APPLICATION NOTE 222-4

GUIDELINES FOR SIGNATURE ANALYSIS
Understanding the Signature Measurement

This Application Note shows how Hewlett-Packard Signature Analyzers take signature measurements. It contains guidelines for controlling the gate through the Start, Stop, and Clock inputs. It shows how measurements of three-state nodes are treated, and more. This information applies to both design and retrofit situations.

FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS PUBLICATION

This application note is aimed at assisting designers, test engineers and others in designing or retrofitting their digital products for Signature Analysis testability and serviceability. It shows how Hewlett-Packard Signature Analyzers take signature measurements. While there are many different ways to control the signature measurement, there are a few common ways that apply to most microprocessor-based products. We've compiled a list of these common measurement control methods in the form of guidelines. These guidelines resulted from suggestions of some of the hundreds who have used signature analysis for the past few years. They are designed to show how to create measurements that result in stable and repeatable signatures, so that correct signatures can be documented in a troubleshooting procedure. When that goal is met, then incorrect, unrepeatable, or unstable signatures measured during troubleshooting, will accurately indicate incorrect or intermittant circuit behavior, allowing fast fault analysis to the component level.

ABOUT OTHER PUBLICATIONS

Application Note 222-0, HP Publication 02-5952-7593, is a complete index to the latest Signature Analysis publications. It lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests, from how to design or retrofit Signature Analysis into digital systems, to the cost reductions that can be expected in production test and field service by doing so. It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

CONTENTS

SECTION PAGE

SECTION 1

Introduction

Contents

This application note explains how Hewlett-Packard Signature Analyzers take signature measurements. Here is what is covered in this note:

1. How the GATE, or measurement cycle, is controlled, including simplified examples that demonstrate basic gating concepts. Some application examples are also given to demonstrate the concepts in a real situation.
2. How data on three-state nodes is interpreted by the signature analyzer during a measurement cycle.
3. How the signature analyzer responds to a reset, and how the HOLD feature works in conjunction with reset.
4. How the signature analyzer is affected by noise on any of its inputs.
5. The signature analyzer specifications.
6. Ways the measurement cycle can be controlled by the product under test, through a combination of hardware and software, during the execution of a circuit stimulus program.

Examples of typical circuit stimulus programs are the subject of additional Case Study notes in this Application Note series. They show typical examples of stimulus program software and the circuits that get stimulated. The Case Studies also contain additional examples of how the GATE is controlled. See Application Note 222-O, "An Index to Signature Analysis Publications," HP Publication 02-5952-7593, for a complete list of current application notes in the AN 222 series.

Organization

Each section of this application note covers a measurement concept by showing simple examples of START, STOP, CLOCK, and DATA waveforms. In some cases, actual application examples will also be shown. The title for each section reflects the way in which most people phrase questions about how the signature analyzer takes a measurement. This organization allows the note to be used as a quick reference guide when a specific question comes up about a measurement concept. But the organization also allows someone unfamiliar with the signature measurement process, to read through it, without having to refer back and forth to different sections.

The simplified examples are intended only to demonstrate and clarify a measurement concept. They do not necessarily reflect an actual measurement. For example, the START, STOP, CLOCK and DATA waveforms are simplified so that they could easily be drawn in diagrams. GATE times were shortened, and the CLOCK is shown with a fixed frequency. However, GATES can be much longer than shown (GATE length is a subject covered in a section of the same name), and the CLOCK need not be constant. The CLOCK can be symmetrical as shown in most of the examples, but it need not be.

Signature Analyzer Characteristics

Here is a list of the four major characteristics of Hewlett-Packard Signature Analyzers that determine the signature value resulting from a signature measurement cycle. Other signature analyzers may or may not get the same signatures, depending on how they treat these and other characteristics. The first two characteristics are shown in the Figures 2.1 and 2.2. The third and fourth characteristics are covered in the remaining sections of this application note.

1. The internal 16-bit signature register feedback taps.
2. The signature display characters.
3. The gating characteristics.
4. The input logic thresholds.

The Signature Register

Figure 2.1 shows the model for the 16-bit internal signature register including the four feedback tap positions. It also shows how the signature analyzer compresses a node's waveform into a four digit signature display. More on the operation of the shift register and its accuracy of error detection can be found in Application Note 222-2.

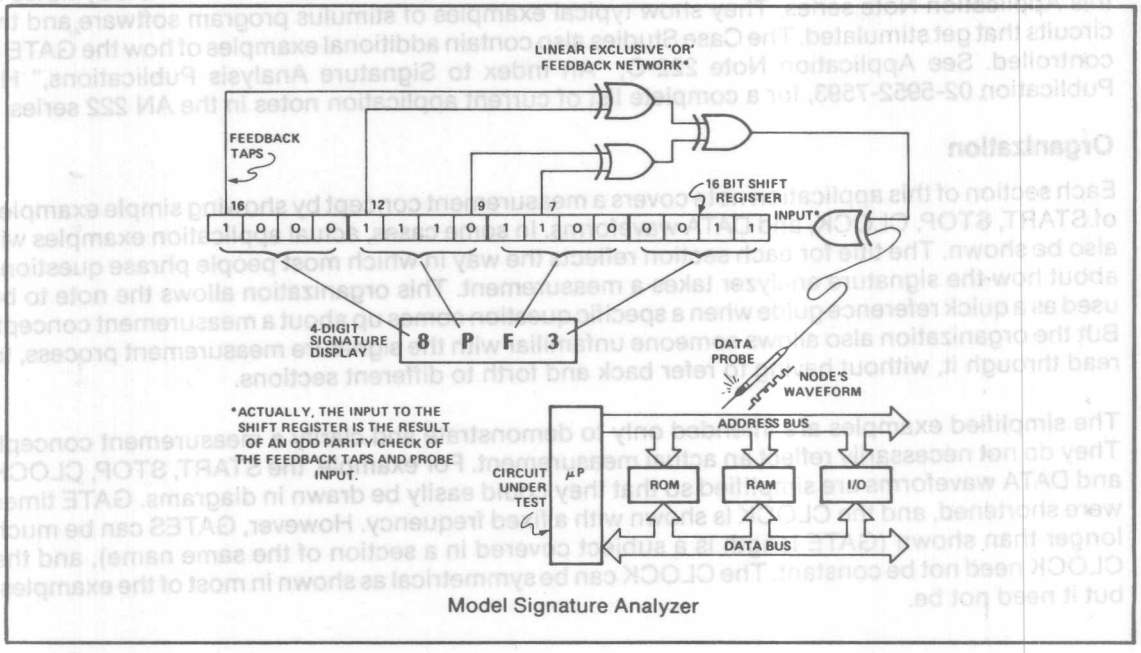
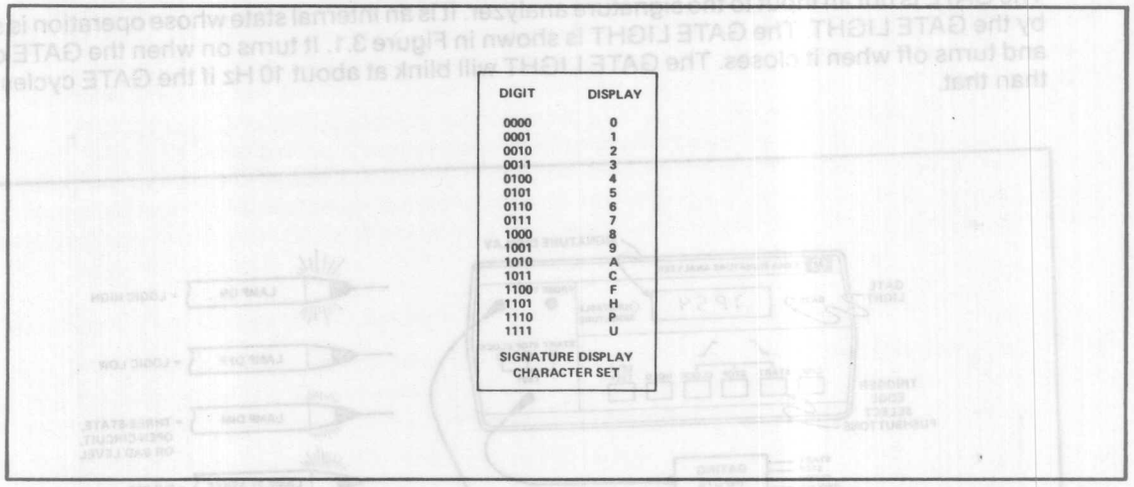


Figure 2.1

The Signature Display Character Set

Figure 2.2 shows the sixteen characters that form each of the four digits of the signature display. The character set has been changed from standard hexidecimal, becasue the signature analyzer uses a seven-segment display. The seven-segment display was chosen to make the characters as large as economically possible, for easy readability during troubleshooting. The new character set eliminates the confusion between the small letter b and the number 6, or the capital letter B and the number 8. During troubleshooting, measured signatures are compared against documented ones. There is no diagnostic information in the signature itself. So it doesn't matter WHAT the characters are, as long as they are EASY to compare. The new character set makes the signature instantly recognizable for fast troubleshooting.



DIGIT	DISPLAY
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	C
1100	F
1101	H
1110	P
1111	U

SIGNATURE DISPLAY CHARACTER SET

Figure 2.2

Gating Characteristics and Input Thresholds

The gating characteristics are covered in sections three through ten of this application note. Input threshold is the subject of section 11 and Appendix A.

SECTION 3

Basic Gate Operation

Definition of the Gate

The measurement cycle of the signature analyzer is controlled through the GATE. When the GATE is open, the analyzer is taking a new signature measurement through the DATA probe input. The signature from the previous measurement cycle remains displayed. When the GATE closes, the analyzer stops taking the new signature and then displays it.

The GATE is not an input to the signature analyzer. It is an internal state whose operation is shown by the GATE LIGHT. The GATE LIGHT is shown in Figure 3.1. It turns on when the GATE opens, and turns off when it closes. The GATE LIGHT will blink at about 10 Hz if the GATE cycles faster than that.

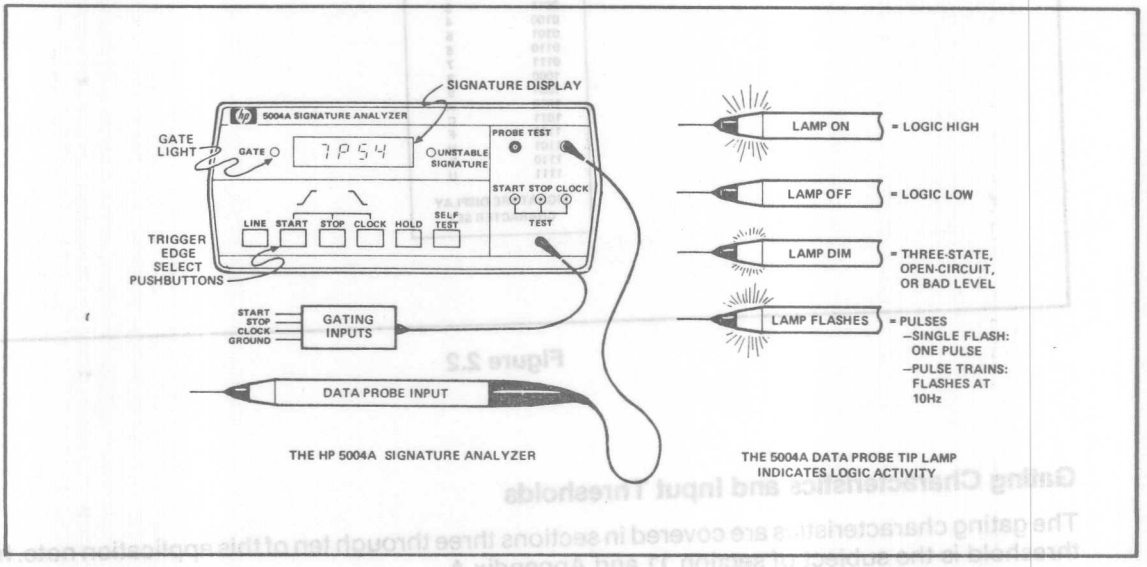
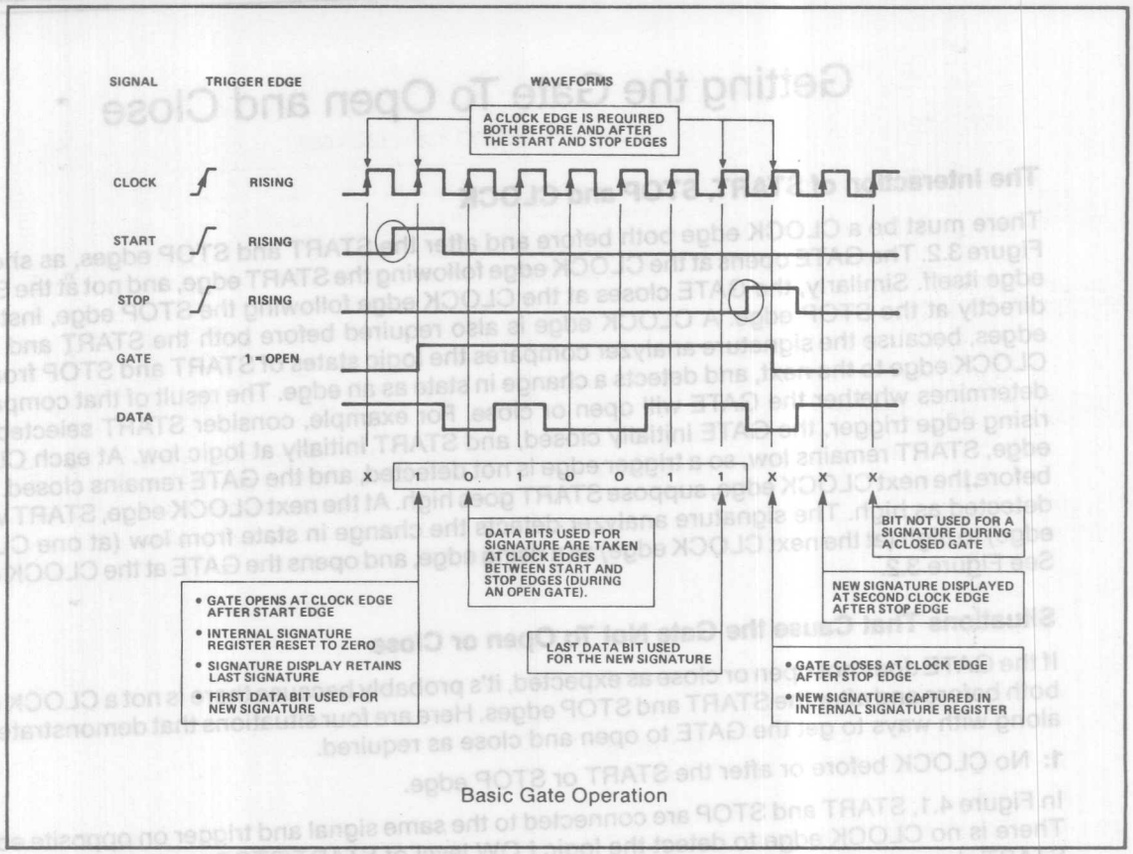


Figure 3.1

The START, STOP, CLOCK and DATA Inputs

The GATE is controlled by the three gating inputs: START, STOP and CLOCK. Basic GATE control is shown in Figure 3.2. START and STOP are used to open and close the GATE at selected trigger edges. The trigger edges for START, STOP and CLOCK are selected by pushbuttons on the instrument, and can be either the rising or falling edge of the input signal, in any combination. From now on, any reference to a START, STOP or CLOCK edge will mean the edge as selected by these pushbuttons.



Basic Gate Operation

Figure 3.2

The CLOCK synchronizes the analyzer to the device being tested. The logic level of the waveform measured by the DATA probe input is sampled at the CLOCK edge and is ignored at all other times. These logic levels are used as data bits that are compressed into the signature. The START and STOP inputs are also synchronously detected by the CLOCK. The GATE opens at the CLOCK edge following the START edge, and not at the START edge itself. Similarly, the GATE closes at the CLOCK edge following the STOP edge, instead of directly at the STOP edge.

First and Last Data Bits Sampled For a Signature

The first DATA bit sampled for use in a signature occurs coincident with the GATE opening at the CLOCK edge following the START edge. The last DATA bit used occurs at the CLOCK edge PRIOR to the STOP edge. In other words, DATA is sampled at every CLOCK edge BETWEEN the START and STOP edges. DATA is not used for a signature when the GATE closes at the CLOCK edge following the STOP edge. The signature is displayed at the SECOND CLOCK edge after the STOP edge.

There are many different ways to control the GATE of the signature analyzer. For instance, both START and STOP can be connected to the same signal or different ones. And they can be triggered on the same or different edges. START usually opens the GATE and STOP closes it, but the GATE can toggle open and closed as well. The GATE can be as long or as short as required. START and STOP can even have multiple edges. The remaining guidelines explain each one of these characteristics and more.

Getting the Gate To Open and Close

The Interaction of START, STOP and CLOCK

There must be a CLOCK edge both before and after the START and STOP edges, as shown in Figure 3.2. The GATE opens at the CLOCK edge following the START edge, and not at the START edge itself. Similarly, the GATE closes at the CLOCK edge following the STOP edge, instead of directly at the STOP edge. A CLOCK edge is also required before both the START and STOP edges, because the signature analyzer compares the logic states of START and STOP from one CLOCK edge to the next, and detects a change in state as an edge. The result of that comparison determines whether the GATE will open or close. For example, consider START selected for a rising edge trigger, the GATE initially closed, and START initially at logic low. At each CLOCK edge, START remains low, so a trigger edge is not detected, and the GATE remains closed. Now, before the next CLOCK edge, suppose START goes high. At the next CLOCK edge, START will be detected as high. The signature analyzer detects the change in state from low (at one CLOCK edge) to high (at the next CLOCK edge) as a rising edge, and opens the GATE at the CLOCK edge. See Figure 3.2.

Situations That Cause the Gate Not To Open or Close

If the GATE does not open or close as expected, it's probably because there is not a CLOCK edge both before and after the START and STOP edges. Here are four situations that demonstrate this, along with ways to get the GATE to open and close as required.

1: No CLOCK before or after the START or STOP edge.

In Figure 4.1, START and STOP are connected to the same signal and trigger on opposite edges. There is no CLOCK edge to detect the logic LOW level of START/STOP, so the rising edge for START does not open the GATE. Similarly the falling edge for STOP won't close the GATE if it starts open.

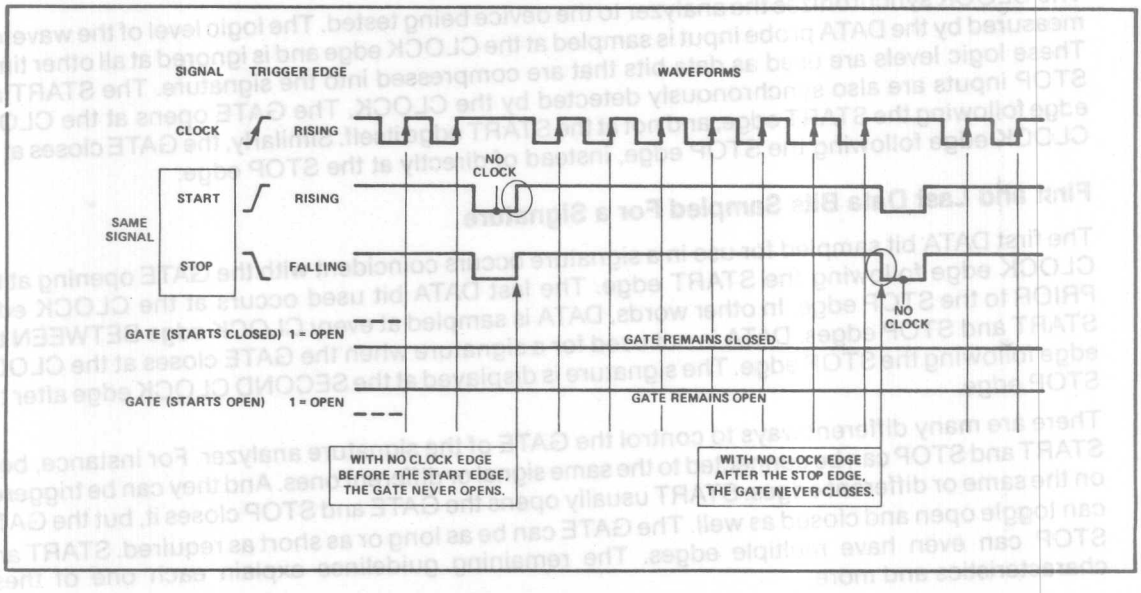
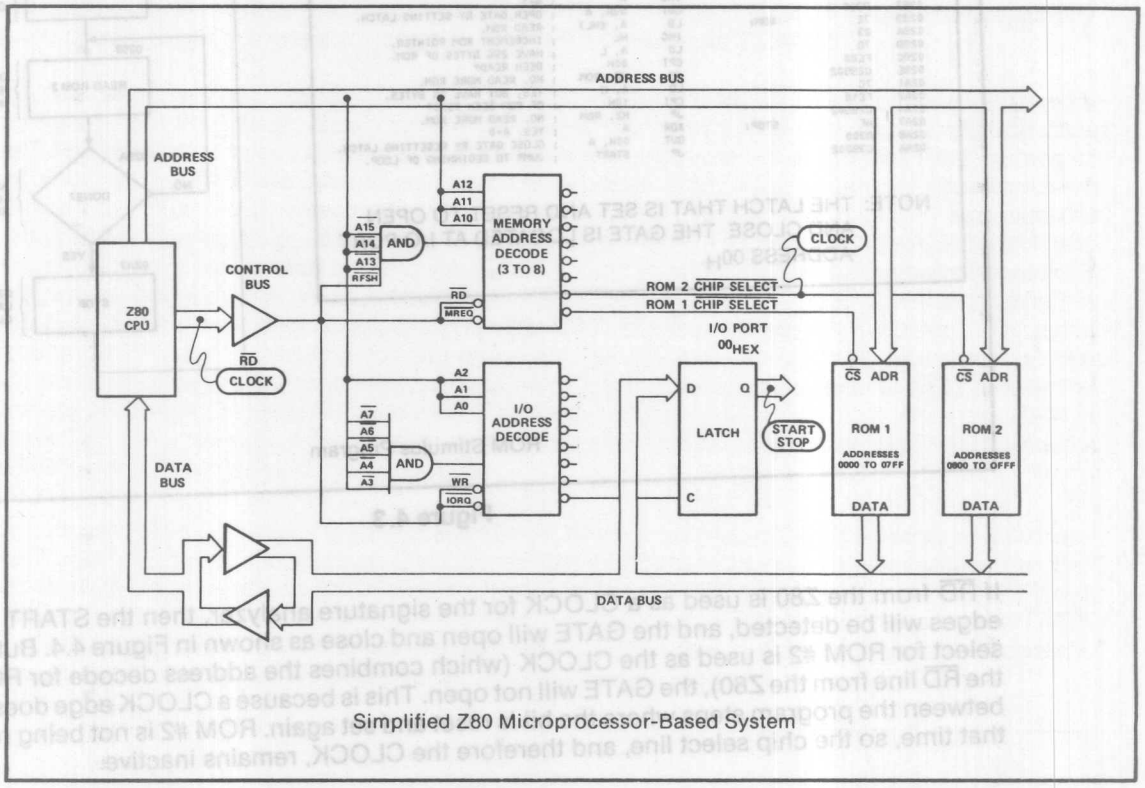


Figure 4.1

Here's an example of this situation from an actual application. In the Zilog Z80 microprocessor-based system of Figure 4.2, START and STOP are connected to a bit in a latch addressed as an I/O port. START is selected to trigger on a rising edge, and STOP triggers on a falling edge. The START and STOP edges are generated by a combination of this hardware, and the execution of the ROM stimulus program of Figure 4.3. The bit in the latch is set to logic one at the beginning of the program to open the GATE, and reset to logic zero at the end to close the GATE.



Simplified Z80 Microprocessor-Based System

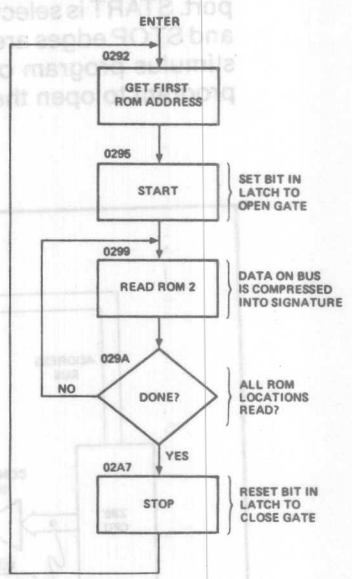
Figure 4.2

The ROM stimulus program of Figure 4.3 is stored in ROM #1, and is used to exercise ROM #2. This allows signature analysis to be used to troubleshoot ROM #2 and associated circuits such as address decoders. The program simply reads all locations of ROM #2 onto the data bus. Before the first ROM #2 location is read, the program sets the bit in the latch to open the GATE. Now all ROM #2 data that is read onto the bus will be used for a signature. After reading all ROM #2 locations, the program resets the bit to close the GATE, then returns to the beginning to open the GATE and start over.

Z80 ASSEMBLY CODE

HEX ADRS	CONTENTS	LABEL	INSTRUCTION		COMMENTS
0292	210008	START:	LD	HL, 0800	; H-L POINTS TO START OF ROM.
0295	AF		XOR	A	; A=0
0296	3C		INC	A	; A=1
0297	D300		OUT	00H, A	; OPEN GATE BY SETTING LATCH.
0299	7E	ROM:	LD	A, (HL)	; READ ROM.
029A	23		INC	HL	; INCREMENT ROM POINTER.
029B	7D		LD	A, L	; HAVE 256 BYTES OF ROM.
029C	FE00		CPI	00H	; BEEN READ?
029E	C29902		JP	NZ, ROM	; NO. READ MORE ROM.
02A1	7C		LD	A, H	; YES, BUT HAVE 2K BYTES.
02A2	FE10		CPI	10H	; OF ROM BEEN READ?
02A4	C29902		JP	NZ, ROM	; NO. READ MORE ROM.
02A7	AF	STOP:	XOR	A	; YES. A=0
02A8	D300		OUT	00H, A	; CLOSE GATE BY RESETTING LATCH.
02AA	C39202		JP	START	; JUMP TO BEGINNING OF LOOP.

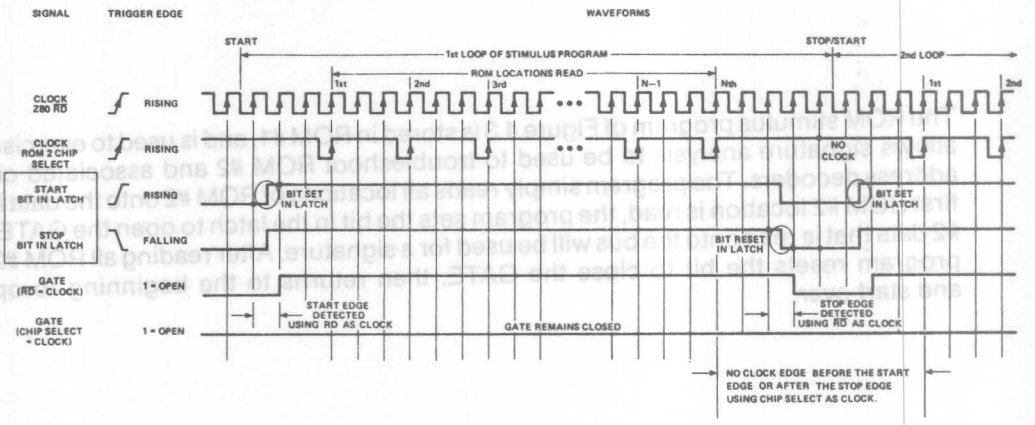
NOTE: THE LATCH THAT IS SET AND RESET TO OPEN AND CLOSE THE GATE IS LOCATED AT I/O PORT ADDRESS 00_H.



ROM Stimulus Program

Figure 4.3

If $\overline{RD}$ from the Z80 is used as a CLOCK for the signature analyzer, then the START and STOP edges will be detected, and the GATE will open and close as shown in Figure 4.4. But if the chip select for ROM #2 is used as the CLOCK (which combines the address decode for ROM #2 and the $\overline{RD}$ line from the Z80), the GATE will not open. This is because a CLOCK edge does not occur between the program steps where the bit is reset and set again. ROM #2 is not being read during that time, so the chip select line, and therefore the CLOCK, remains inactive.



ROM Stimulus Program Timing Diagram Showing Start and Stop Edge Detection

Figure 4.4

Why not connect CLOCK to $\overline{RD}$ instead of the chip select for ROM #2 and be satisfied? The answer depends on the data that should, and should not be, clocked into the signature analyzer. In this example, a $\overline{RD}$ CLOCK occurs every time the processor reads data from ROM #2. Therefore, if a signature is taken on the data bus, then it will be composed of data from ROM #2 as intended. But a $\overline{RD}$ CLOCK also occurs every time the processor fetches instructions from ROM #1 during program execution. Therefore, the signature is also composed of data from ROM #1 as well as ROM #2. This is satisfactory as long as the contents of ROM #1 is known to be good, or can be verified by some other means such as FREERUN. (The contents of ROM #1 had better be good or else the program will not execute properly, and will not exercise ROM #2.)

What if it's necessary to clock data into the signature analyzer **only** when a device is exercised by the program (in order to keep data from being entered into the signature analyzer during any other time)? For instance, imagine that this example program was used to stimulate RAM, instead of ROM, and that the RAM was accessed by both the processor and another device. This occurs in a CRT terminal where the processor stores characters in RAM, and the CRT regularly accesses the RAM to put the characters on the screen. This interaction of the processor and the CRT with RAM, is random with respect to each other. If the CLOCK is connected to $\overline{RD}$, then the data sampled by the signature analyzer will be a combination of data as accessed by the processor (during the stimulus program), and data as accessed by the CRT (during refresh of the screen). Since this combination of data is random, the signature will be unstable. To get a stable signature, the CLOCK is connected to the chip select of the RAM. The chip select combines the $\overline{RD}$ line with the processor's address decode for the RAM. The chip select line is active only when the RAM is accessed by the processor. Now the signature analyzer will sample data from the RAM only while the RAM is read during the stimulus program.

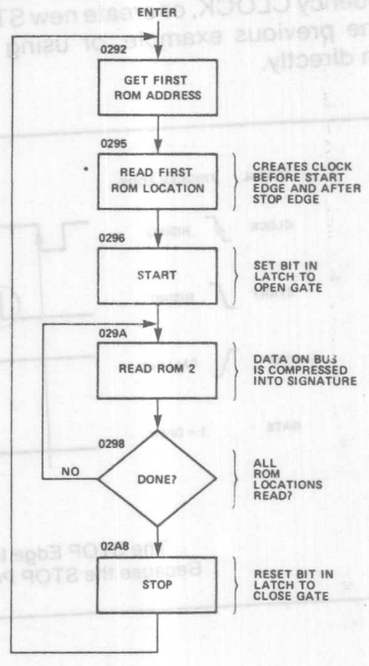
In the original ROM example, if the chip select for ROM #2 is used as a CLOCK, then only data from ROM #2 will be used for a signature. Data from ROM #1 will be ignored. However, when the chip select for ROM #2 is used as a CLOCK, then the START and STOP edges are not detected. This is because the chip select is not active, and therefore there's no CLOCK edge, between the START and STOP edge generation. To create the required CLOCK edge, a minor step could be added to the program that reads the first ROM location as shown in Figure 4.5.

Z80 ASSEMBLY CODE

	HEX ADRS	CONTENTS	LABEL	INSTRUCTION		COMMENTS
	0292	210008	START:	LD	HL, 0800	; H-L POINTS TO START OF ROM.
*	0295	7E		LD	A, (HL)	; READ FIRST ROM LOCATION.
	0296	AF		XOR	A	; A=0
	0297	3C		INC	A	; A=1
	0298	D300		OUT	00H, A	; OPEN GATE BY SETTING LATCH.
	029A	7E	ROM:	LD	A, (HL)	; READ ROM.
	029B	23		INC	HL	; INCREMENT ROM POINTER.
	029C	7D		LD	A, L	; HAVE 256 BYTES OF ROM.
	029D	FE00		CPI	00H	; BEEN READ?
	029F	C29A02		JP	NZ, ROM	; NO. READ MORE ROM.
	02A2	7C		LD	A, H	; YES, BUT HAVE 2K BYTES.
	02A3	FE10		CPI	10H	; OF ROM BEEN READ?
	02A5	C29A02		JP	NZ, ROM	; NO. READ MORE ROM.
	02A8	AF	STOP:	XOR	A	; YES. A=0
	02A9	D300		OUT	00H, A	; CLOSE GATE BY RESETTING LATCH.
	02AB	C39202		JP	START	; JUMP TO BEGINNING OF LOOP.

NOTE: THE LATCH THAT IS SET AND RESET TO OPEN AND CLOSE THE GATE IS LOCATED AT I/O PORT ADDRESS 00_H.

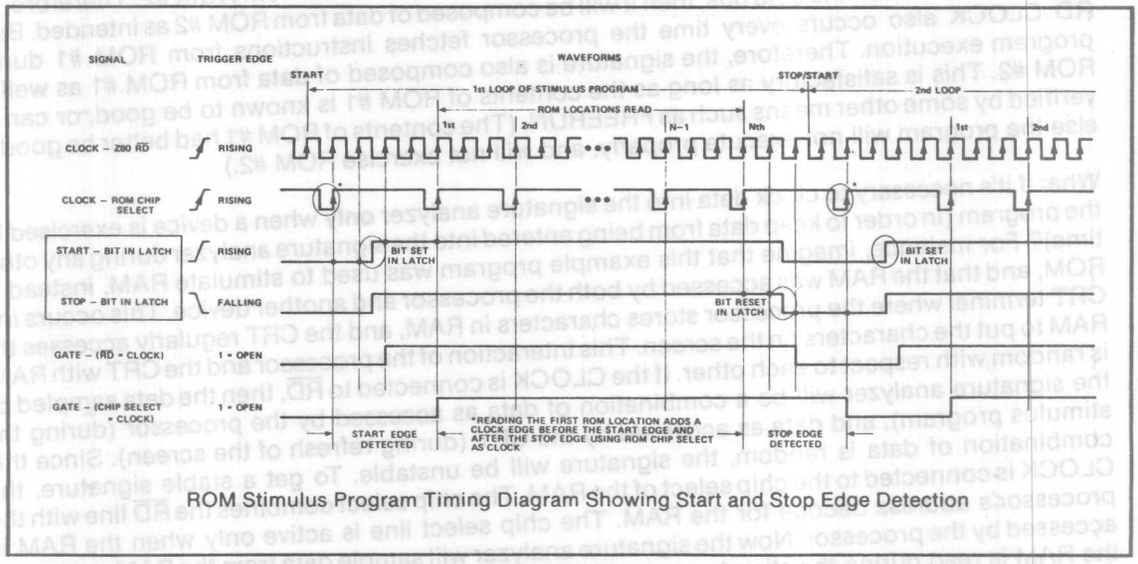
* EXTRA PROGRAM STEP



ROM Stimulus Program With Step Added To Detect the Start and Stop Edges

Figure 4.5

This inserts a clock pulse into the CLOCK waveform as shown in Figure 4.6. Now a CLOCK edge occurs both before and after the START and STOP edges, and the GATE opens and closes as required.

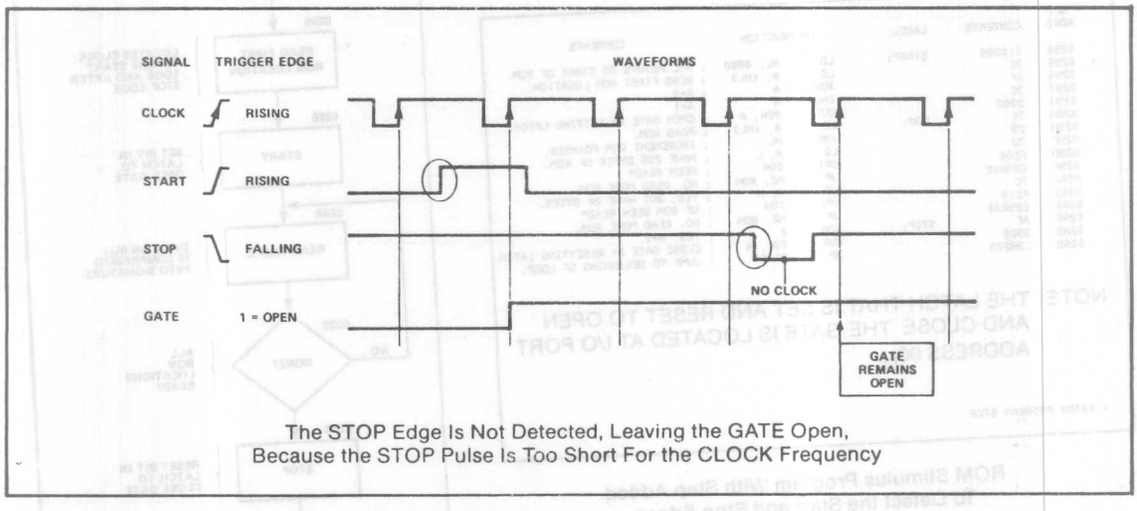


ROM Stimulus Program Timing Diagram Showing Start and Stop Edge Detection

Figure 4.6

2: The START or STOP pulse is too short for the CLOCK frequency.

In the example of Figure 4.7, the STOP pulse is too short for the slower CLOCK frequency. The GATE remains open because the STOP edge is not detected. This can occur when START and STOP are connected to address decodes, and pulsed at the beginning and end of the stimulus program. If a CLOCK is chosen that is slower than the pulses, such as a UART's transmitter clock input, then the pulses will remain undetected. To get the GATE to open and close, choose a higher frequency CLOCK, or create new START and STOP edges by controlling a bit in a latch as shown in the previous example, or using the address decode pulses to set and reset a flip-flop or latch directly.

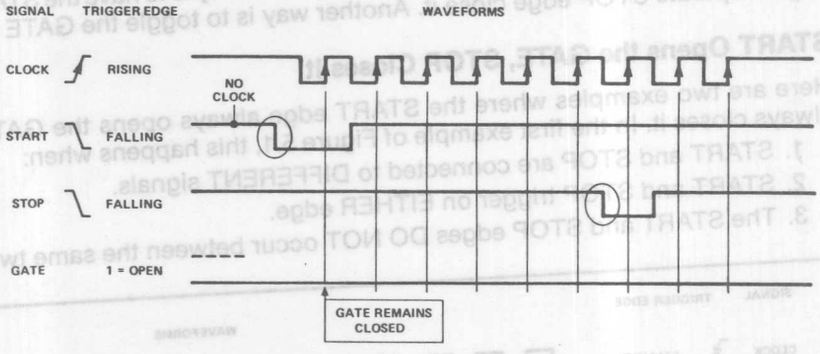


The STOP Edge Is Not Detected, Leaving the GATE Open,
Because the STOP Pulse Is Too Short For the CLOCK Frequency

Figure 4.7

3: The CLOCK turns on after the START edge.

In figure 4.8, the START edge is not detected because it occurs before the first CLOCK edge. This can happen in board test systems, where START is the signal from the board tester that turns on the CLOCK to the board under test. Here, selecting the other edge for START will allow the GATE to open. Choosing a CLOCK that is running before the START edge will also ensure its detection.

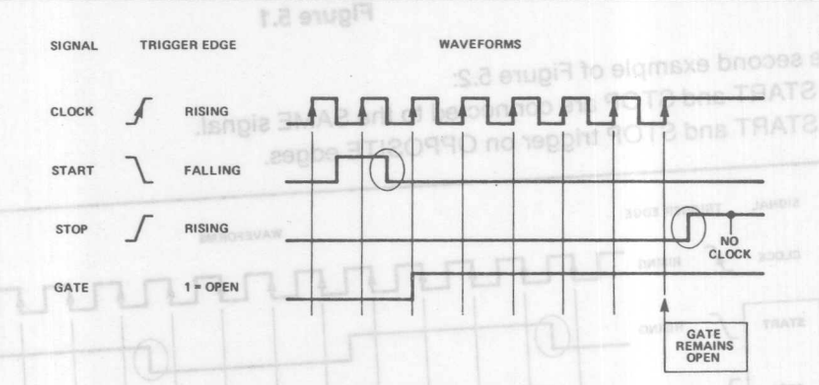


The START Edge Is Not Detected, and the GATE Never Opens, Because the CLOCK Turns On After the START Edge

Figure 4.8

4: The CLOCK turns off before the STOP edge.

In Figure 4.9, the STOP edge is not detected because it occurs after the last CLOCK edge. In this example, STOP is the terminal count output of a counter. When the terminal count occurs, the counter's clock is turned off. Connecting CLOCK to the counter's clock will cause the STOP edge to remain undetected. To ensure detection, the CLOCK should be moved to a clock that runs after the STOP edge.



The STOP Edge Is Not Detected, Leaving the GATE Open, Because the CLOCK Turns Off Before the STOP Edge

Figure 4.9

Two Ways To Control The Gate

There are two basic ways to control the GATE. One way is to have the START edge open the GATE and a separate STOP edge close it. Another way is to toggle the GATE open and closed.

START Opens the GATE, STOP Closes It

Here are two examples where the START edge always opens the GATE, and the STOP edge always closes it. In the first example of Figure 5.1, this happens when:

1. START and STOP are connected to DIFFERENT signals.
2. START and STOP trigger on EITHER edge.
3. The START and STOP edges DO NOT occur between the same two CLOCK edges.

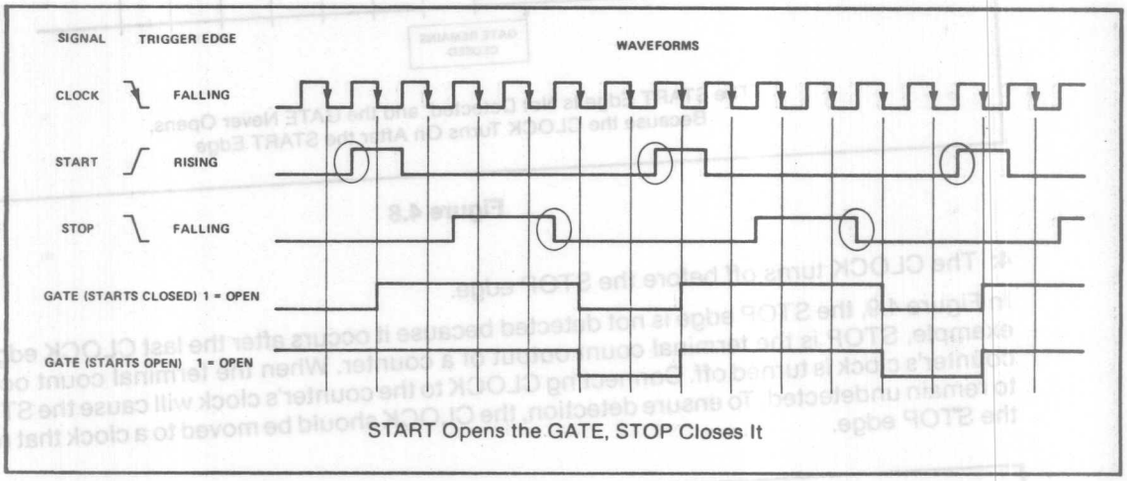


Figure 5.1

In the second example of Figure 5.2:

1. START and STOP are connected to the SAME signal.
2. START and STOP trigger on OPPOSITE edges.

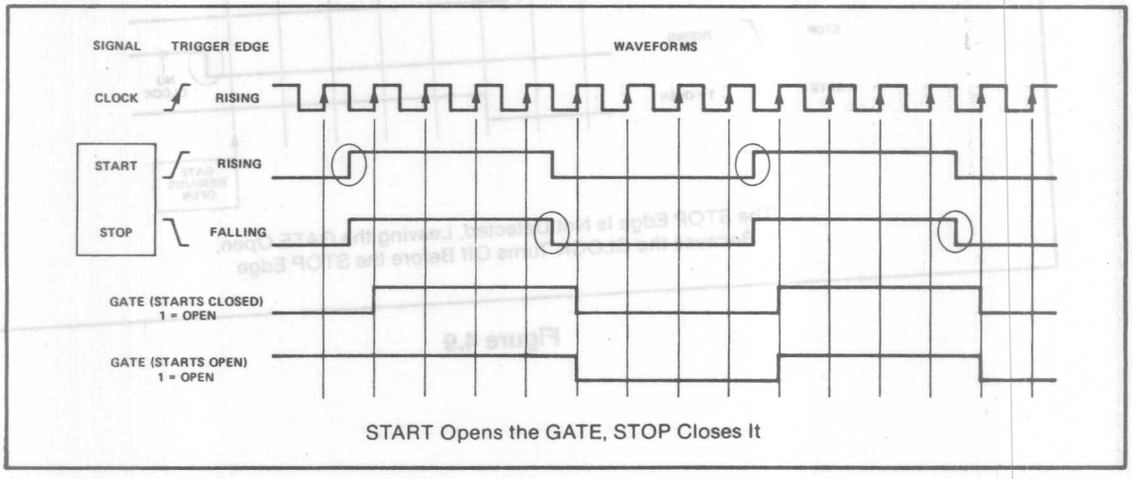


Figure 5.2

The GATE Toggles Open and Closed

In these examples, the GATE toggles because the START and STOP edges occur between the same two CLOCK edges. In the first example of Figure 5.3:

1. START and STOP are connected to the SAME signal.
2. START and STOP both trigger on the SAME edge.

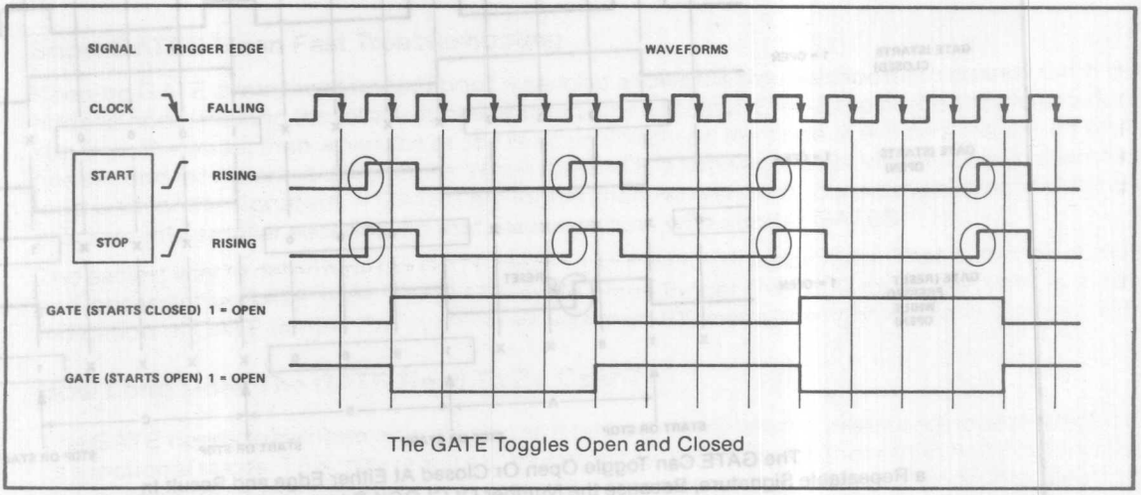


Figure 5.3

In the second example of Figure 5.4:

1. START and STOP are connected to DIFFERENT signals.
2. START and STOP trigger on EITHER edge.
3. The START and STOP edges occur between the SAME two CLOCK edges.

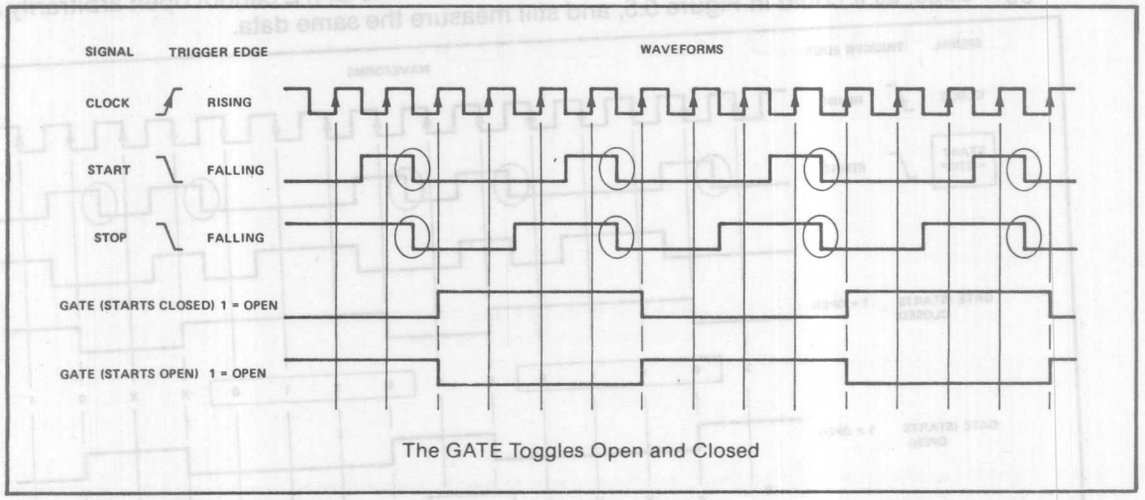
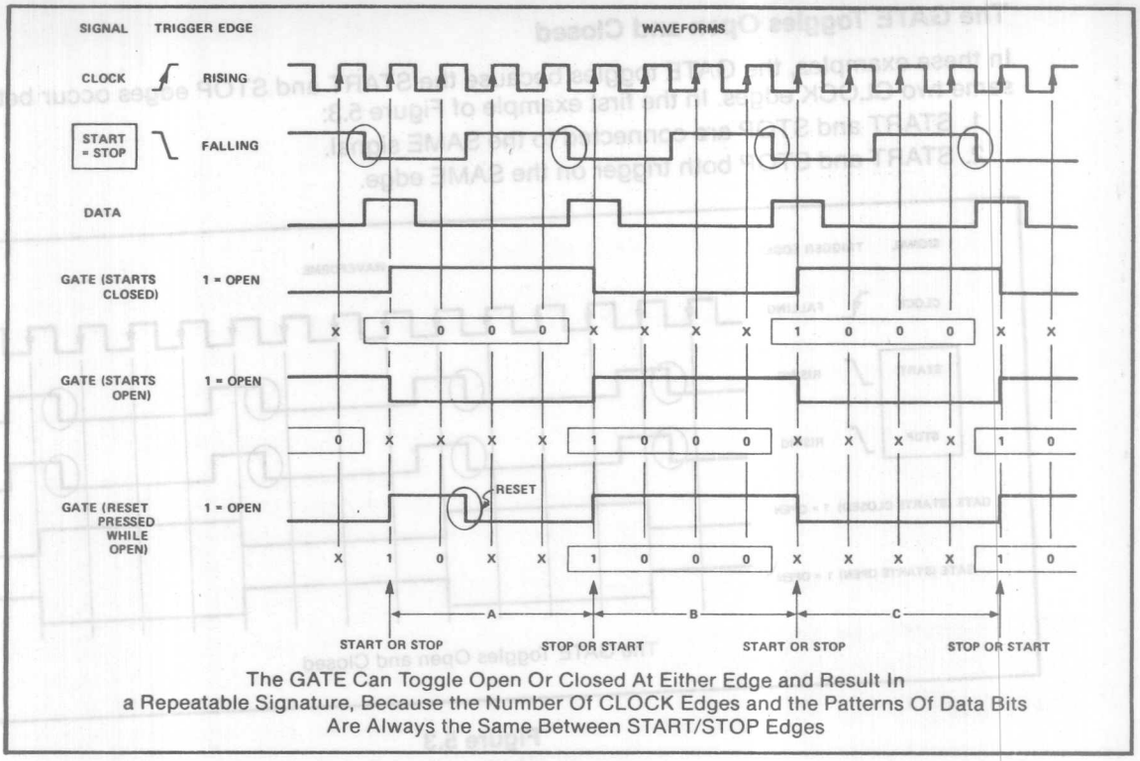


Figure 5.4

When toggling the GATE, keep the same number of CLOCK edges, and the same pattern of DATA bits, between alternate trigger edges, so that the GATE can open arbitrarily at either edge. This insures repeatable signatures. Figures 5.5 through 5.7 show this.

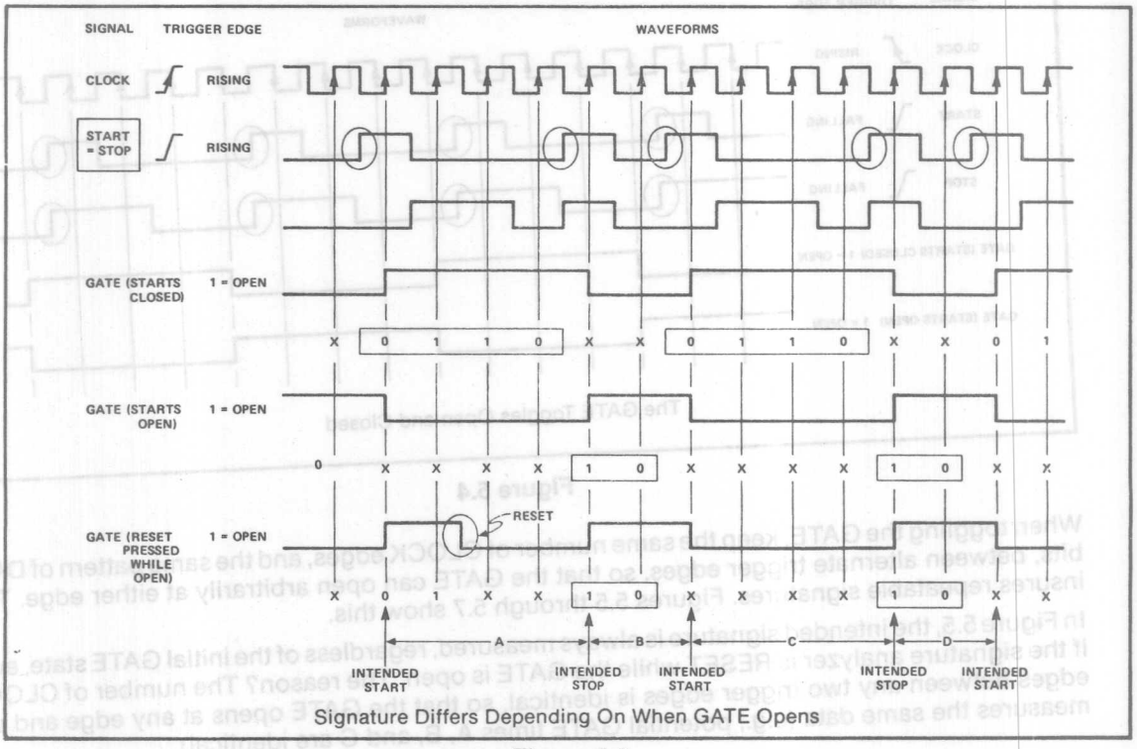
In Figure 5.5, the intended signature is always measured, regardless of the initial GATE state, even if the signature analyzer is RESET while the GATE is open. The reason? The number of CLOCK edges between any two trigger edges is identical, so that the GATE opens at any edge and still measures the same data (e.g., potential GATE times A, B, and C are identical).



The GATE Can Toggle Open Or Closed At Either Edge and Result In a Repeatable Signature, Because the Number Of CLOCK Edges and the Patterns Of Data Bits Are Always the Same Between START/STOP Edges

Figure 5.5

In Figure 5.6, the GATE is intended to be open during times A and C. However, if the GATE starts open (which can occur if RESET is pressed during A or C), then the GATE will continue to toggle at each new trigger edge, but will be open when it should be closed and vice versa. This will result in signatures taken during times B and D when the GATE is intended to be closed. The reason? The times between any two trigger edges are not the same. The GATE cannot open arbitrarily at any trigger edge, as it could in Figure 5.5, and still measure the same data.

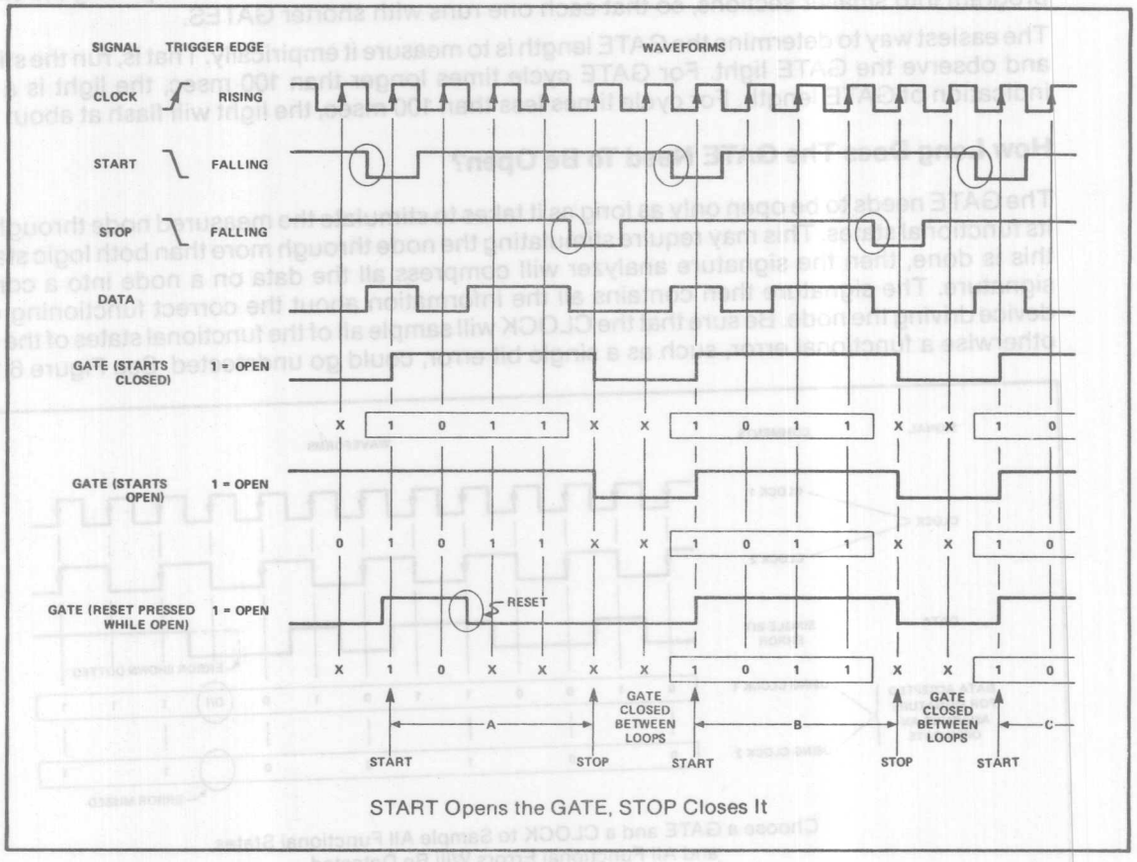


Signature Differs Depending On When GATE Opens

Figure 5.6

There are at least two ways to make the GATE open and close when intended. One way is to keep START and STOP on the same signal and continue to toggle the GATE. However, create an edge at the beginning of the stimulus program or time period of interest to open the GATE. Do not create another edge (for STOP) at the end of the program as in Figure 5.6. Instead, cause the program to return to the beginning where the same edge will then close the GATE for the next execution of the loop. The GATE will then continue to toggle as shown in Figure 5.5.

Another way is to have START open the GATE and STOP close it, as shown in Figure 5.7. Do this by connecting START and STOP to separate signals. Create one START edge at the beginning of the stimulus program or time period of interest to open the GATE. Create one STOP edge to close the GATE at the end of the program. Even pressing RESET will not change the times the GATE will open. This is especially useful if the GATE must be closed for an extended period between loops of the stimulus.



START Opens the GATE, STOP Closes It

Figure 5.7

Gate Lengths

Short GATES Mean Fast Troubleshooting

Keeping GATE cycle times under about ½ second allows the troubleshooter to quickly jump from node to node, making the total troubleshooting time very fast. Even if the probe should accidently slip from the node, then when the probe is again solidly on the node, it will be a maximum wait of one second for the correct signature. When the GATE is two seconds or longer, it could seem like a long wait for the signature. If GATES are longer than two seconds, consider splitting the stimulus program into smaller sections, so that each one runs with shorter GATES.

The easiest way to determine the GATE length is to measure it empirically. That is, run the stimulus and observe the GATE light. For GATE cycle times longer than 100 msec, the light is a direct indication of GATE length. For cycle times less than 100 msec, the light will flash at about 10 Hz.

How Long Does The GATE Need To Be Open?

The GATE needs to be open only as long as it takes to stimulate the measured node through all of its functional states. This may require stimulating the node through more than both logic states. If this is done, then the signature analyzer will compress all the data on a node into a compact signature. The signature then contains all the information about the correct functioning of the device driving the node. Be sure that the CLOCK will sample all of the functional states of the node, otherwise a functional error, such as a single bit error, could go undetected. See Figure 6.1.

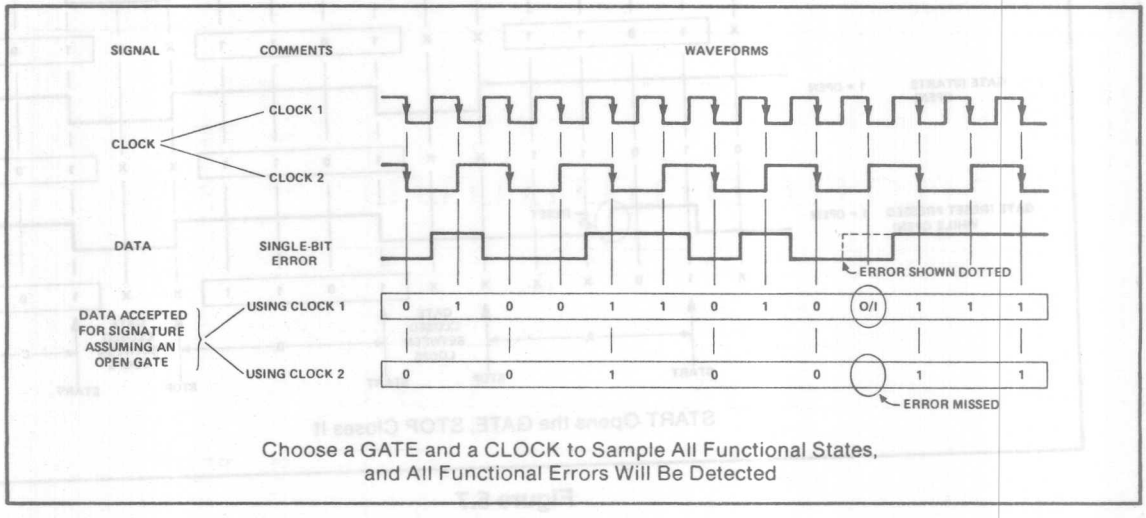


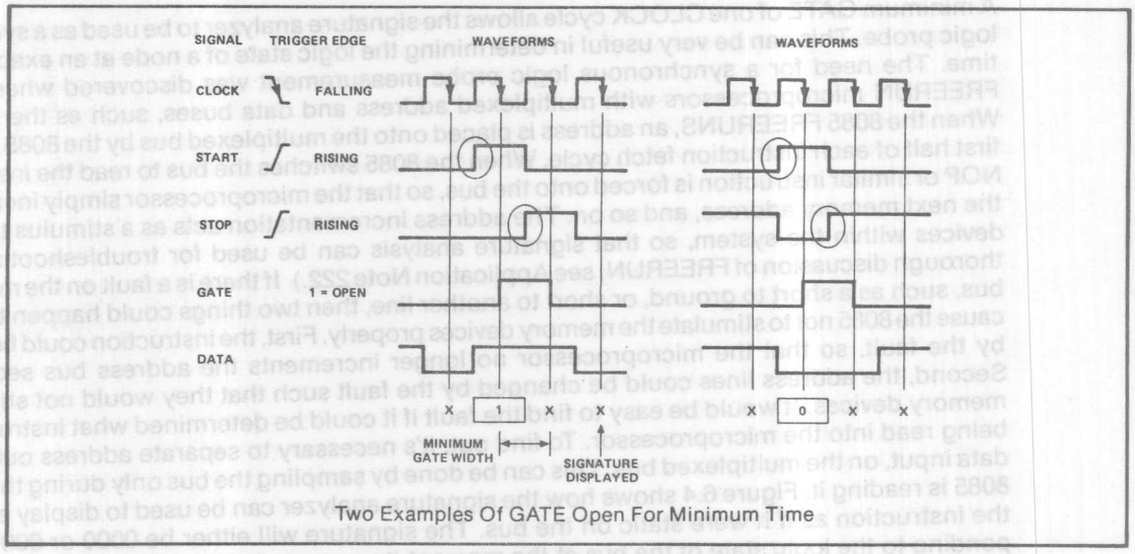
Figure 6.1

Single bit errors are guaranteed to result in a signature change for that node, as long as the error bit has been clocked into the signature analyzer. Multiple bit errors will result in a signature change for that node with a probability of 99.998%. In other words, functional failures in the device driving the node will be detected with a probability of 99.998%, worst case, assuming the error bits have been clocked into the signature analyzer, and that the device has been exercised through all of its functions.

It's unnecessary to artificially lengthen the GATE to increase the accuracy of error detection. The signature analyzer's accuracy of error detection is 100% for 16 or less data bits clocked into the analyzer. For more than 16 bits, the accuracy remains at 99.998%, worst case. This means that GATES can be as short or as long as necessary. Application Note 222-2 contains a complete discussion of the accuracy of error detection.

Minimum and Maximum Timing For An Open GATE

At least one CLOCK edge must occur between the START edge and the STOP edge, in order to open then close the GATE. The GATE opens for one CLOCK edge, while one data bit is clocked into the signature analyzer. The new signature will be displayed at the SECOND CLOCK edge after the STOP edge (i.e., after the GATE closes). See Figure 6.2.



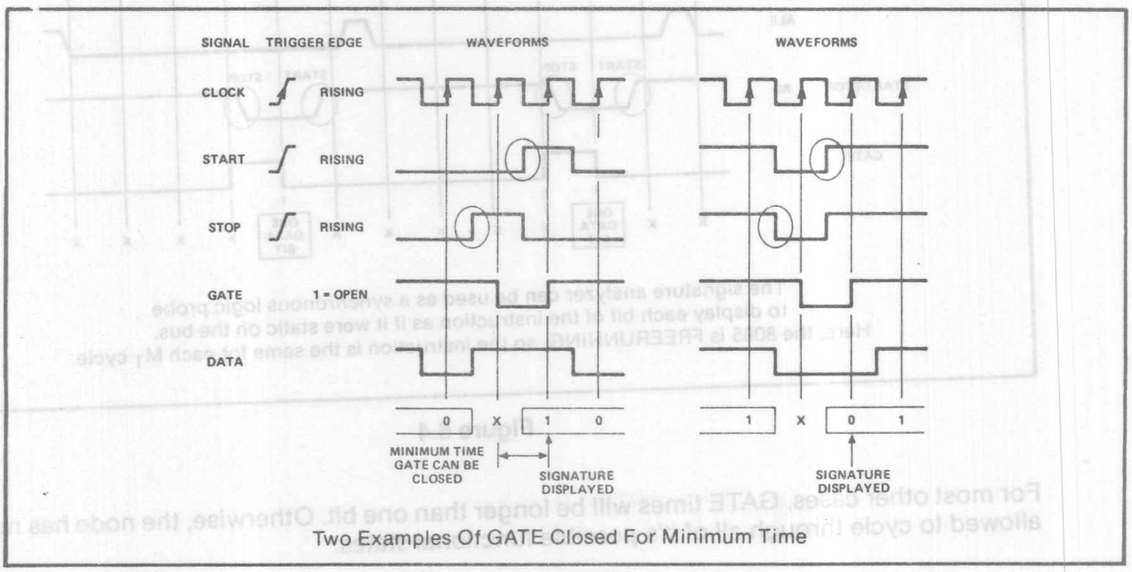
Two Examples Of GATE Open For Minimum Time

Figure 6.2

Any number of CLOCK edges can occur between a START edge and a STOP edge. There is no upper limit. Even the number of possible signatures (65,536) is not an upper limit. The signature analyzer will continue to gather a signature as long as the GATE is open. However, the new signature will not be displayed until the SECOND CLOCK edge after the STOP edge (i.e., not until the GATE closes).

Minimum and Maximum Timing For A Closed GATE

At least one CLOCK edge must occur between the last STOP edge and the next START edge, in order to close then open the GATE. The GATE will close for that CLOCK edge, and the data bit at that edge will not be used for the signature. The signature will be displayed at the SECOND CLOCK edge after the STOP edge (i.e., when the GATE opens again). See Figure 6.3.



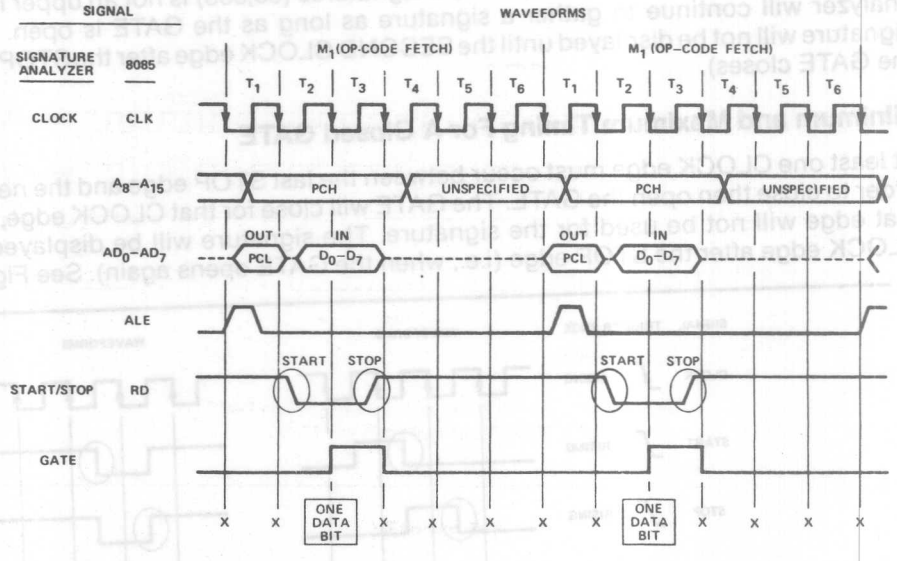
Two Examples Of GATE Closed For Minimum Time

Figure 6.3

Any number of CLOCK edges can occur between the last STOP edge and the next START edge. There is no upper limit. The signature analyzer will not gather a signature as long as the GATE is closed. The last signature will be displayed at the SECOND CLOCK edge after the STOP edge (i.e., after the GATE closed), and will continue to be displayed as long as the GATE is closed.

Creating A Synchronous Logic Probe

A minimum GATE of one CLOCK cycle allows the signature analyzer to be used as a synchronous logic probe. This can be very useful in determining the logic state of a node at an exact instant in time. The need for a synchronous logic probe measurement was discovered when trying to FREERUN microprocessors with multiplexed address and data buses, such as the Intel 8085. When the 8085 FREERUNS, an address is placed onto the multiplexed bus by the 8085 during the first half of each instruction fetch cycle. When the 8085 switches the bus to read the instruction, a NOP or similar instruction is forced onto the bus, so that the microprocessor simply increments to the next memory address, and so on. The address incrementation acts as a stimulus to memory devices within the system, so that signature analysis can be used for troubleshooting. (For a thorough discussion of FREERUN, see Application Note 222.) If there is a fault on the multiplexed bus, such as a short to ground, or short to another line, then two things could happen that would cause the 8085 not to stimulate the memory devices properly. First, the instruction could be changed by the fault, so that the microprocessor no longer increments the address bus sequentially. Second, the address lines could be changed by the fault such that they would not stimulate all memory devices. It would be easy to find the fault if it could be determined what instruction was being read into the microprocessor. To find out, it's necessary to separate address output, from data input, on the multiplexed bus. This can be done by sampling the bus only during the time the 8085 is reading it. Figure 6.4 shows how the signature analyzer can be used to display each bit of the instruction as if it were static on the bus. The signature will either be 0000 or 0001, corresponding to the logic state of the bus at the moment it was sampled.



The signature analyzer can be used as a synchronous logic probe to display each bit of the instruction as if it were static on the bus. Here, the 8085 is FREERUNNING, so the instruction is the same for each M_1 cycle.

Figure 6.4

For most other cases, GATE times will be longer than one bit. Otherwise, the node has not been allowed to cycle through all of it's possible functional states.

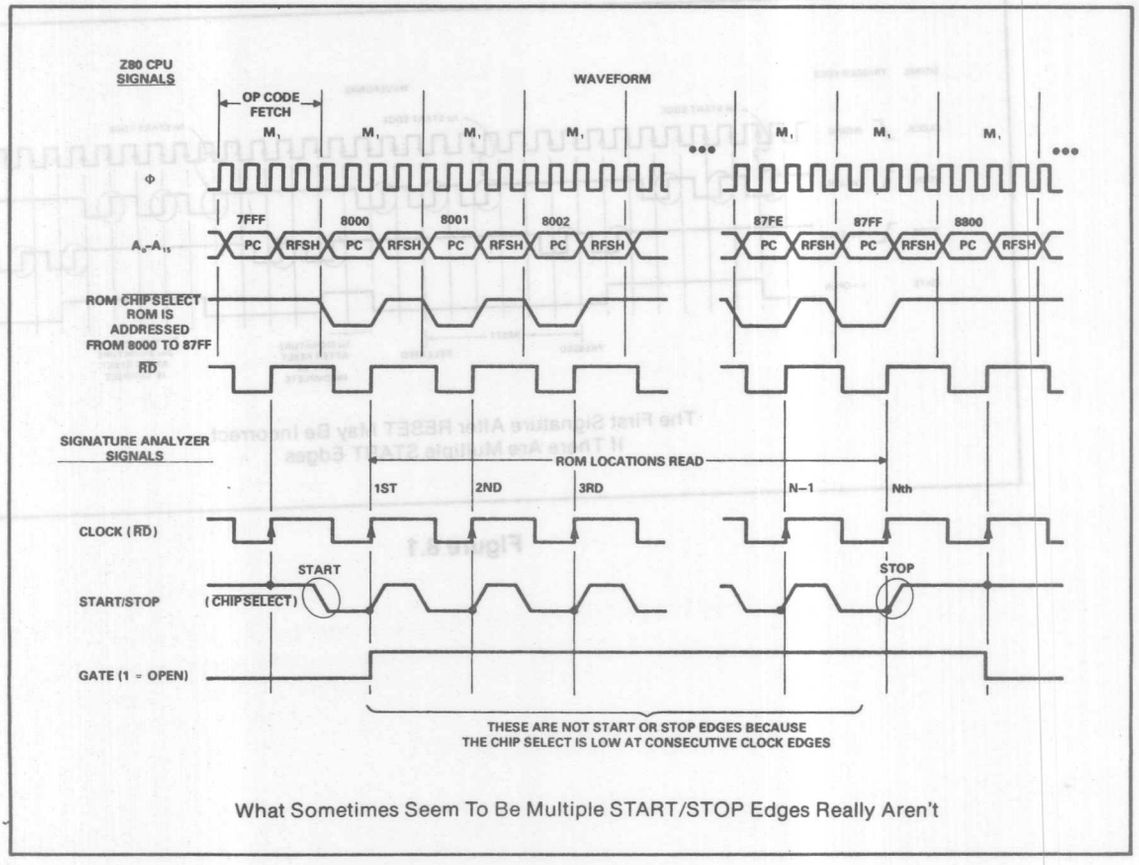
Multiple START and STOP Edges

If there is another START edge between any START edge and a STOP edge, then there are multiple START edges. Similarly, if there is at least one additional STOP edge between any STOP edge and a START edge, then there are multiple STOP edges. This assumes two things. First, that the START and STOP inputs are connected to separate signals. Second, that the START and STOP edges are detected by the CLOCK. The next paragraph shows how to determine if the second condition is true. If both conditions are true, then the first START edge after the GATE closes will open the GATE. Further START edges have no effect. The first STOP edge after the GATE opens will close the GATE. Further STOP edges have no effect. The GATE opens again at the first (next) START edge after the GATE closes. Resetting the signature analyzer, or using the HOLD feature, may affect the operation of the GATE. See Section Eight.

Are There Really Multiple START or STOP Edges?

Signals sometimes seem to have more than one edge when they really don't. For example, when FREERUNNING a microprocessor such as the Zilog Z80, START and STOP are frequently connected to ROM chip selects or their equivalents. This creates a GATE that's open only while ROM data is on the bus, allowing ROM's and associated circuits to be verified and troubleshot.

The chip selects become active at each address, and then inactive again during address changes. This creates multiple edges that at first seem to be multiple START and STOP edges. But when $\overline{RD}$ is used as a CLOCK, only one START and STOP edge actually occur. See Figure 7.1.



What Sometimes Seem To Be Multiple START/STOP Edges Really Aren't

Figure 7.1

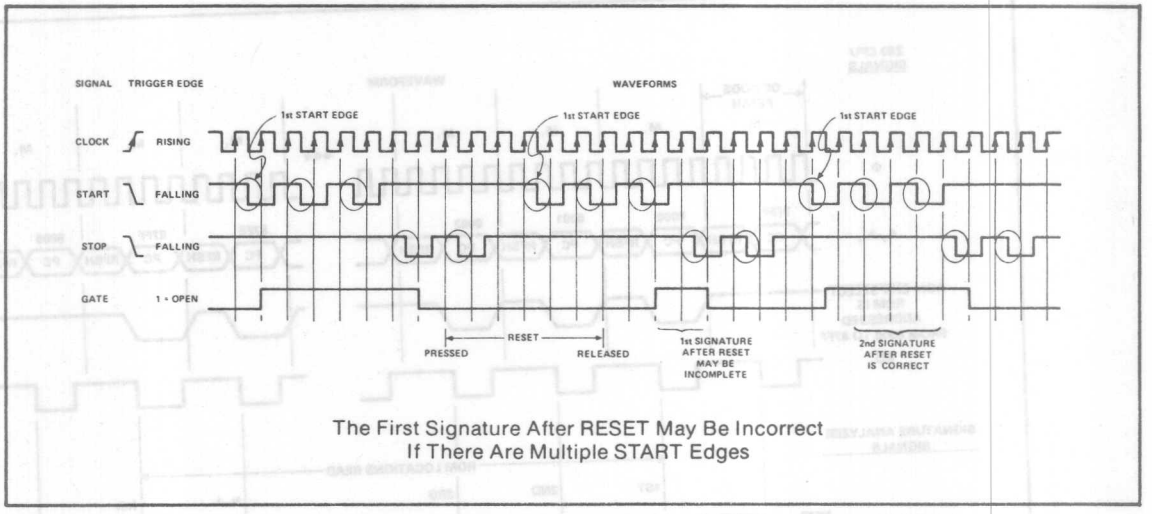
Reset and Hold

Resetting The Signature Analyzer

The HP Model 5004A Signature Analyzer is reset by pressing the RESET button on the DATA probe. The signature analyzer is usually reset for one of the following reasons:

1. It's a convenient way to cause the signature analyzer to display the V_{CC} (all ones) signature without touching the DATA probe to V_{CC}.
2. It's the only way to get the signature analyzer to take another signature while it's in the HOLD mode.
3. It's a way to reset the display and close the GATE, just after placing the DATA probe on a node, so that the next signature displayed is the correct one. This is usually done when GATE cycle times are long.

When there are multiple START edges, the first signature displayed after reset may not be the correct one. As shown in Figure 8.1, when the signature analyzer is RESET, the GATE closes, and will remain closed, until the next START edge. If the GATE opens again at the first of multiple START edges, then the first signature will be a correct (complete) one. If the GATE reopens at one of the multiple START edges other than the first one, then the first signature will be a partial (incorrect) one. In either case, the second signature after RESET will always be correct (complete), assuming the GATE repeats and the signature analyzer is not in the HOLD mode.



The First Signature After RESET May Be Incorrect
If There Are Multiple START Edges

Figure 8.1

The HOLD Feature

The HOLD feature causes the HP 5004A Signature Analyzer to take only ONE signature after RESET. To use the HOLD feature, press the HOLD button IN. Place the DATA probe on the node, and then press RESET for a new signature. The next signature displayed will be held in the display, even if the probe is removed from the node. This feature is handy when the probe must be removed from the node temporarily to see the display. This can happen when troubleshooting a product that's in an awkward location. The HOLD feature also makes it convenient to document signatures in the troubleshooting procedure. After the signature is captured in the display, the probe can be set down so that the signature can be written. HOLD can even be used to capture signatures of single-cycle events that occur when the product is first turned on. Note that a second signature is not taken in the HOLD mode. If the HOLD feature is intended to be used, then START signals must have only one edge. If this is done, then the "held" signature will always be correct (complete).

Getting Correct, Repeatable, and Stable Signatures

While creating an SA circuit stimulus, a few circuit conditions can cause unstable or unrepeatable signatures. This section outlines their causes and how to recognize them. The remaining guidelines show how to eliminate them from the SA stimulus design so that incorrect signatures measured during troubleshooting accurately indicate a fault.

Correct and Incorrect Signatures

Correct signatures are defined as the signatures documented in the troubleshooting procedure for a product. An incorrect signature is defined as a signature displayed on the signature analyzer, that doesn't match the documented one for the node being probed. If the correct signatures are repeatable, then measuring an incorrect signature during troubleshooting will accurately indicate an incorrect waveform for that node. This allows the troubleshooter to quickly back-trace through the circuit following incorrect waveforms to the faulty node. The fault is found at the point where back-tracing further results in correct signatures again. For instance, if the signature for an output of a device is incorrect, signatures are taken on the inputs for that device. If the input signatures are correct, then the fault has been isolated to the output node. If any input signature is incorrect, back-tracing continues along that signal path.

A troubleshooter is concerned only if the signatures are correct or incorrect. He must believe that the signatures documented for the product are correct. That is, he assumes that a known good product will yield all of the signatures as documented in the troubleshooting procedure. The person that creates and checks the signature analysis stimulus is concerned about assuring that signatures to be documented in the troubleshooting procedure are really correct. Correct signatures must be repeatable.

The creator of the stimulus also is concerned about whether all possible product faults will yield incorrect signatures. Two things determine this. First, the accuracy of error detection of the measurement, and second, the extent of the functional stimulus of the product. The signature analysis measurement gives a 99.998% worst-case probability of detecting an error, as long as the error appears in the waveform for that node, and the error bit(s) have been clocked into the signature analyzer. The error will appear in the waveform only if the functional stimulus causes it to be there.

Repeatable and Unrepeatable Signatures

Repeatable signatures are defined as getting the same signature each and every time the signature measurement is made, no matter when it is made, if:

1. Identical nodes in the product are measured.
2. Identical known good products are tried.
3. Identical signature measurements are performed.
4. Identical signature analysis stimulus is run.
5. Identical signature analyzers are used (i.e., signature analyzers with compatible characteristics. See Section 2 for a description of the characteristics.)

Unrepeatable signatures are defined as getting a different signature any time the same node in a known good product is measured, in the same way, when running the same stimulus. One of the goals of the signature analysis circuit stimulus creator, is to eliminate unrepeatable signatures from the product.

Making Sure Signatures Are Repeatable

Unrepeatable signatures usually occur because some circuit element, such as a flip-flop or RAM, has not been initialized before the SA stimulus begins. They also can occur if the GATE toggles open at the wrong time. The following steps will help uncover repeatable signatures in a product. They should be performed to verify ALL signatures in a product BEFORE they are considered correct.

1. Turn the product under test off then on again.
2. Reset the signature analyzer, or touch the DATA probe to ground or V_{CC} before measuring a node.
3. Take signatures in as many other known good products as time and effort permits.

Turning the product under test off then on again determines if any devices are not being initialized before the SA stimulus begins. For example, a flip-flop that is not set or reset at power-on or at the beginning of the SA stimulus loop can cause signatures to differ. RAM also must be initialized with some known pattern before the SA stimulus begins if any nodes associated with that RAM are measured during the execution of the stimulus.

Resetting the signature analyzer determines two things. First, that the GATE toggles open at the intended time. See Section Five. Second, that the node being sampled is always in the third state. If a node is always in the third state, the signature will either be all zeroes, or the V_{CC} signature, depending on the last valid logic state of the previous node measured. If an all zeroes signature has been documented for the three-state node, WITHOUT a notation that the signature could ALSO be the V_{CC} signature, then the signature will appear to be unrepeatable. (This is a special case where the signature is unrepeatable, but can only be one of two possible signatures.) Reset causes the signature analyzer to use a logic one as the last valid logic state. This is the same as if the probe had been touched to V_{CC} before measuring the three-state node. See Sections Eight and Eleven.

Taking signatures in as many known good products as possible determines if all uninitialized devices have been eliminated from the signature measurement. It also determines if all recommended operating conditions of the signature analyzer have been met. Using another signature analyzer can also help determine this. For example, if the specification for the setup time is being violated, but the signature analyzer actually is performing better, then another signature analyzer may help uncover this. See Appendix A for setup and hold times.

Here's an example of a situation that can cause unrepeatable signatures. On the data bus of a FREERUNNING microprocessor system, there may be data from ROM, RAM and I/O devices. The ROM data is always the same, but RAM data will differ each time the system is turned off then on again. This is because the processor has no way to initialize RAM by storing a pattern in it. Therefore, the data from RAM and other uninitialized devices must be eliminated from a signature measurement of the data bus. This will result in the same DATA (ROM data only) being measured each time the GATE is open and therefore results in repeatable signatures. This can be done in two ways. One way is to move START and STOP to ROM chip selects (or their equivalents), to create a GATE that is open only while ROM data is on the bus. Another way to do this is to choose a CLOCK that samples data on the bus only when ROM data is present.

Stable and Unstable Signatures

The definition of stable and unstable signatures depends on the measurement of two signatures in a row. Stable signatures occur when two adjacent signature measurements result in the same signature for all measurements. Unstable signatures occur when the signatures of any two adjacent measurement cycles differ. An unstable signature is indicated by the UNSTABLE SIGNATURE LIGHT. The light turns on for approximately 100 milliseconds whenever a signature taken during ANY GATE cycle differs from the signature taken during the immediately previous GATE cycle. If unstable signatures occur for every GATE cycle, the light will blink at $\leq$10 Hz. Unstable signatures are not of concern while the signature analyzer is in the HOLD mode. This is because only one signature is taken after RESET is pressed. However, the UNSTABLE LIGHT will flash as the signature display changes from 0000 after RESET, to a captured signature other than 0000.

Sometimes the four-digit signature display will indicate an unstable signature by slowly and/or constantly changing, or changing so fast that it becomes unintelligible. However, with fast GATES and an intermittant circuit fault, intermittantly unstable signatures can occur without a noticeable signature display change. In those cases, the UNSTABLE SIGNATURE LIGHT will be the only indication of an unstable signature and the intermittant circuit.

Eliminating Unstable Signatures

Unstable signatures usually occur because the GATE length created by the stimulus is not the same from loop to loop, or the response of the circuit to the stimulus is not the same from loop to loop. For stable and repeatable signatures, the GATE must be open for the same number of CLOCK edges each time it opens. The DATA bits sampled by CLOCK edges during an open GATE must also be identical (same logic levels and time relationship) from one open GATE to the next. While the GATE is closed, the number of CLOCK edges can vary as well as the DATA pattern. See Figure 9.1. To check DMA cycles that transfer the same data each time, connect START to a line that signals when DMA begins. In a Zilog Z80-based system, this might be HOLD ACKNOWLEDGE. This control line from the Z80 signals to an external device, such as a DMA controller, that it now can take control of the bus. Connect STOP to a similar signal that occurs at the end of the DMA cycle. See Figure 9.1.

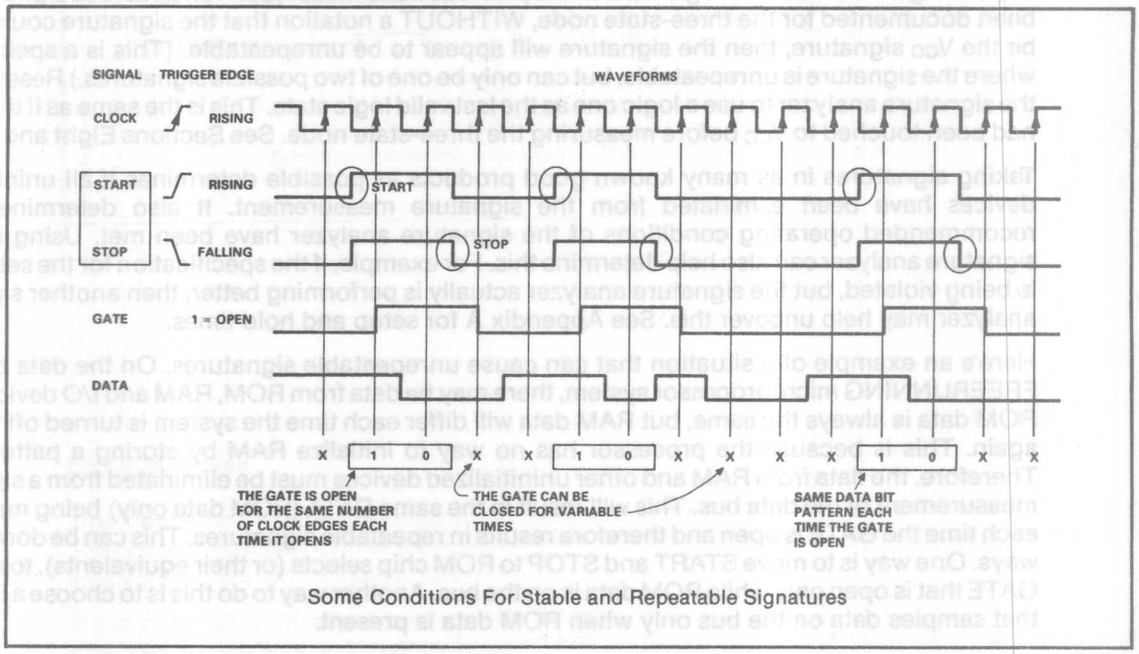


Figure 9.1

Unstable signatures can sometimes be documented for nodes if the signature changes by only a few numbers. For instance, if a signature is unstable, but only changes between two numbers, then both can be documented. Then, either one measured during troubleshooting will indicate correct circuit function, while any other signature will indicate a fault.

SECTION 10

About Noise

Most signal noise does NOT result in incorrect, unstable, or unrepeatable signatures, because the signature analyzer synchronously detects logic state changes on START, STOP and DATA at CLOCK edges. This section shows the effects of two different types of noise, defined below, on each of these four inputs.

Synchronous Noise Definition

Synchronous noise is any noise on a signal that can be predicted with respect to a CLOCK edge. It is usually caused by clocking data synchronously, through a digital system. It consists of momentary transitions, or multiple logic state changes, through the logic threshold opposite to the signal's defined stable state, between two CLOCK edges. These transitions occur after the CLOCK edge that caused them, but are absent at the next CLOCK edge. Here, "at the CLOCK edge" means the signal is at its defined state during the data setup time of the signature analyzer.

Asynchronous Noise Definition

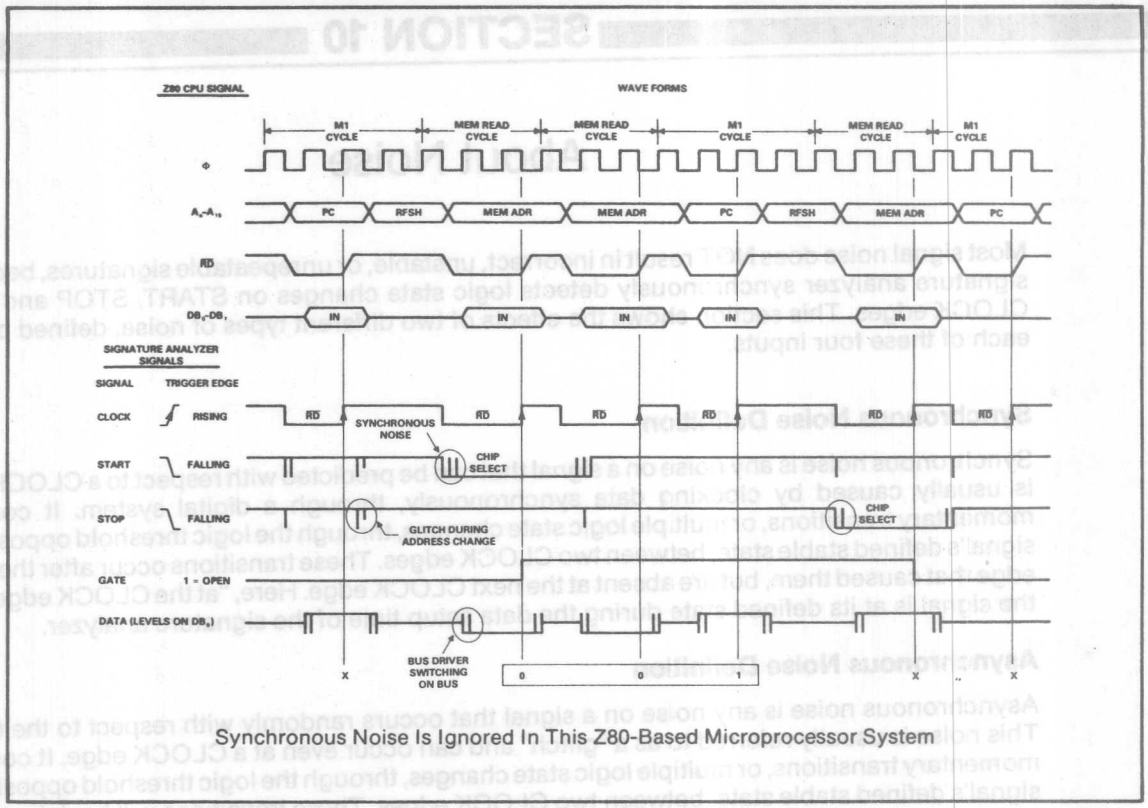
Asynchronous noise is any noise on a signal that occurs randomly with respect to the CLOCK. This noise is usually referred to as a "glitch" and can occur even at a CLOCK edge. It consists of momentary transitions, or multiple logic state changes, through the logic threshold opposite to the signal's defined stable state, between two CLOCK edges. These transitions will be detected as an incorrect logic state only if they occur exactly at the CLOCK edge of the signature analyzer.

Noise On START, STOP and DATA

Synchronous noise on START and STOP is ignored by the signature analyzer, as shown in Figure 10.1. This is a real advantage when using address decoders or chip selects as START and STOP, even though they frequently are noisy during address changes. But since a CLOCK can be used that occurs only after the address bus is stable, the noise does not get detected and will not open or close the gate incorrectly at the noise edges. The gate opens or closes only at a CLOCK edge when the START or STOP edge has been synchronously detected.

Synchronous noise on the DATA input is also ignored. This is an advantage when taking signatures on high-current data bus lines which are noisy while drivers switch on and off the bus. Only valid logic states at CLOCK edges will be used as DATA bits for the signatures.

Asynchronous noise on START, STOP and DATA can cause incorrect, unstable, or unrepeatable signatures if it occurs at the CLOCK edge. This is because it can violate the setup time of the CLOCK as defined in the Recommended Operating Conditions of Appendix A.



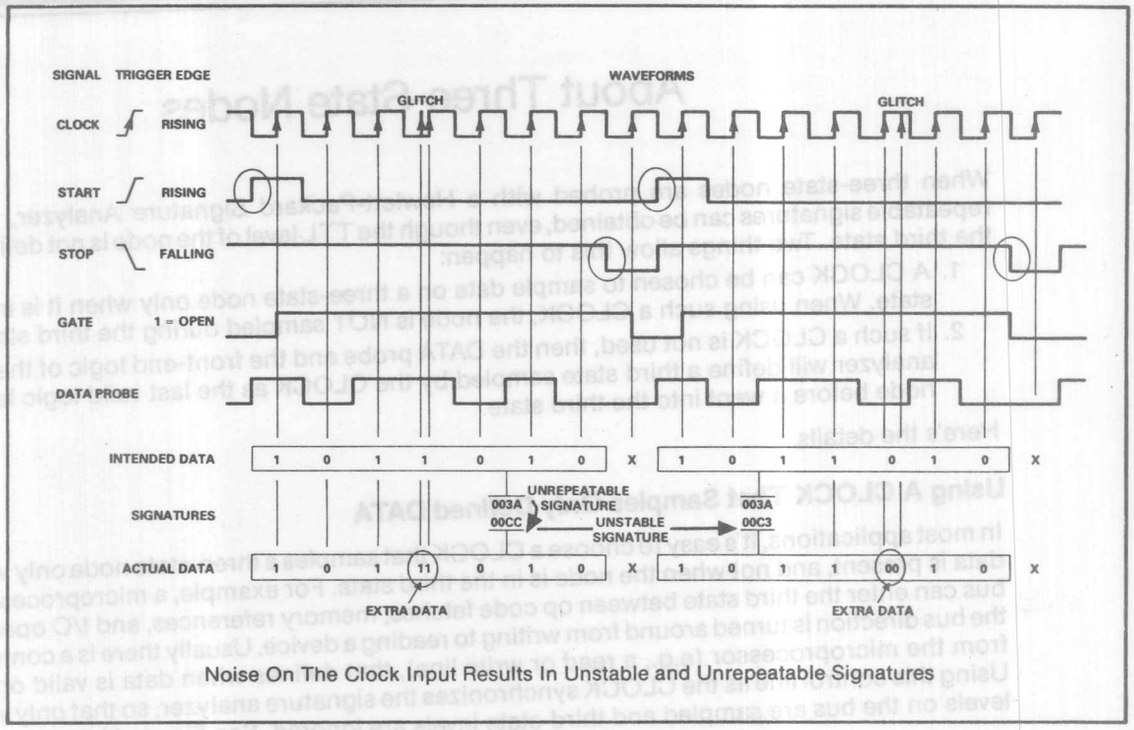
Synchronous Noise Is Ignored In This Z80-Based Microprocessor System

Figure 10.1

CLOCK Noise

Any transition through the single logic threshold of the CLOCK input is defined as a CLOCK edge. Synchronous or asynchronous noise on the CLOCK results in extra CLOCK edges to the signature analyzer, as shown in Figure 10.2.

If the extra CLOCK edges occur while the gate is closed, then the signature may or may not be affected depending on the activity on START. If the extra CLOCK edges do not detect a START edge, then the gate will remain closed and the signature will be correct. However, if the extra CLOCK edges do accidentally detect a START edge, then the gate will open early. The gate length will then vary resulting in an unstable or unrepeatable signature. If the extra CLOCK edges occur while the gate is open, the length of the gate will change, resulting in unstable or unrepeatable signatures.



Noise On The Clock Input Results In Unstable and Unrepeatable Signatures

Figure 10.2

Ground Noise

The ground leads of the signature analyzer should be connected to circuit grounds that are free of excessive noise. Normally only the GATING POD ground lead need be connected to the circuit under test. However, when measuring high-frequency DATA (usually above 5 MHz), an additional ground lead connected to the DATA probe ground point will help reduce any noise induced through a long ground lead path.

About Three-State Nodes

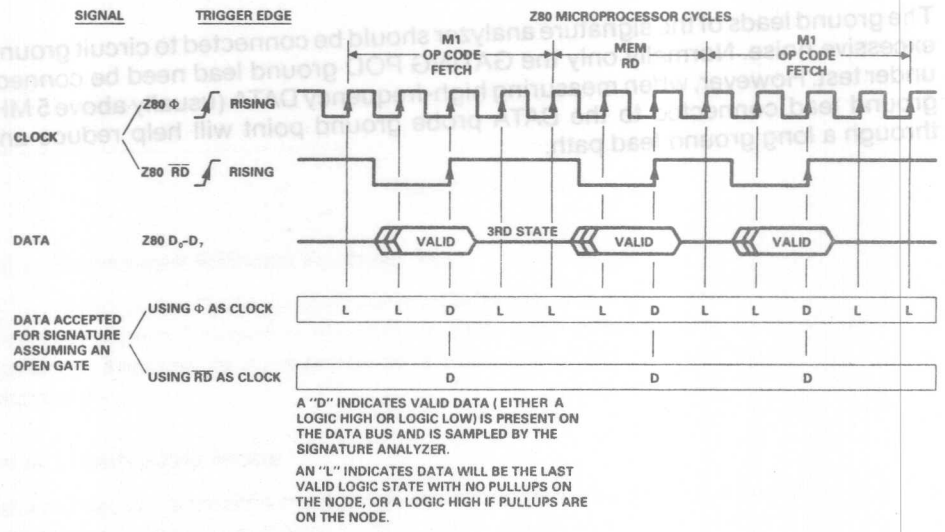
When three-state nodes are probed with a Hewlett-Packard Signature Analyzer, stable and repeatable signatures can be obtained, even though the TTL level of the node is not defined during the third state. Two things allow this to happen:

1. A CLOCK can be chosen to sample data on a three-state node only when it is in a defined state. When using such a CLOCK, the node is NOT sampled during the third state.
2. If such a CLOCK is not used, then the DATA probe and the front-end logic of the signature analyzer will define a third state sampled by the CLOCK as the last valid logic level on the node before it went into the third state.

Here's the details.

Using A CLOCK That Samples Only Defined DATA

In most applications, it's easy to choose a CLOCK that samples a three-state node only when valid data is present, and not when the node is in the third state. For example, a microprocessor's data bus can enter the third state between op code fetches, memory references, and I/O operations as the bus direction is turned around from writing to reading a device. Usually there is a control output from the microprocessor (e.g., a read or write line), that defines when data is valid on the bus. Using this control line as the CLOCK synchronizes the signature analyzer, so that only valid logic levels on the bus are sampled and third-state levels are ignored. See Figure 11.1.



A Clock Can Be Chosen To Sample DATA On a Three-State Node Only When It Is In a Defined State

Figure 11.1

Using A CLOCK That Samples A Node During The Third State

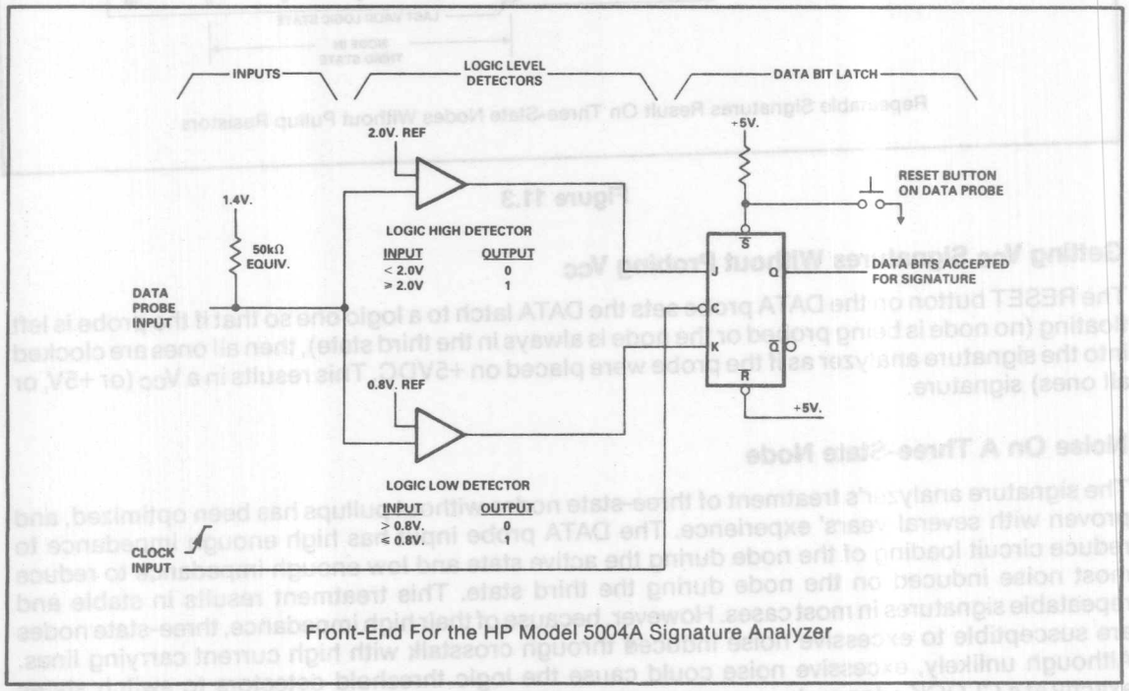
Sometimes it's necessary to use a CLOCK that samples a node when it's in the third state. This can happen when a synchronous CLOCK is not available or when a node remains in the third state for a long time. The signature then depends on whether or not there's an external pullup (or pulldown), on the node.

Signatures For Three-State Nodes Without Pullups

Stable and repeatable signatures can be obtained on three-state nodes without pullups using a Hewlett-Packard Signature Analyzer. A pullup does not need to be added to the node during system design, nor does a pullup need to be added to the DATA probe during troubleshooting. This is because the signature analyzer will continue to use the last valid logic state as a DATA bit when the node enters the third state. There are three parts to the signature analyzer that allow this to happen:

1. An internal reference inside the DATA probe pulls the node to 1.4 volts during the third state.
2. Dual threshold logic level detectors that define logic LOW as 0.8 volts or less, logic HIGH as 2.0 volts or greater, and anything in between LOW and HIGH as the third state.
3. A clocked DATA bit latch.

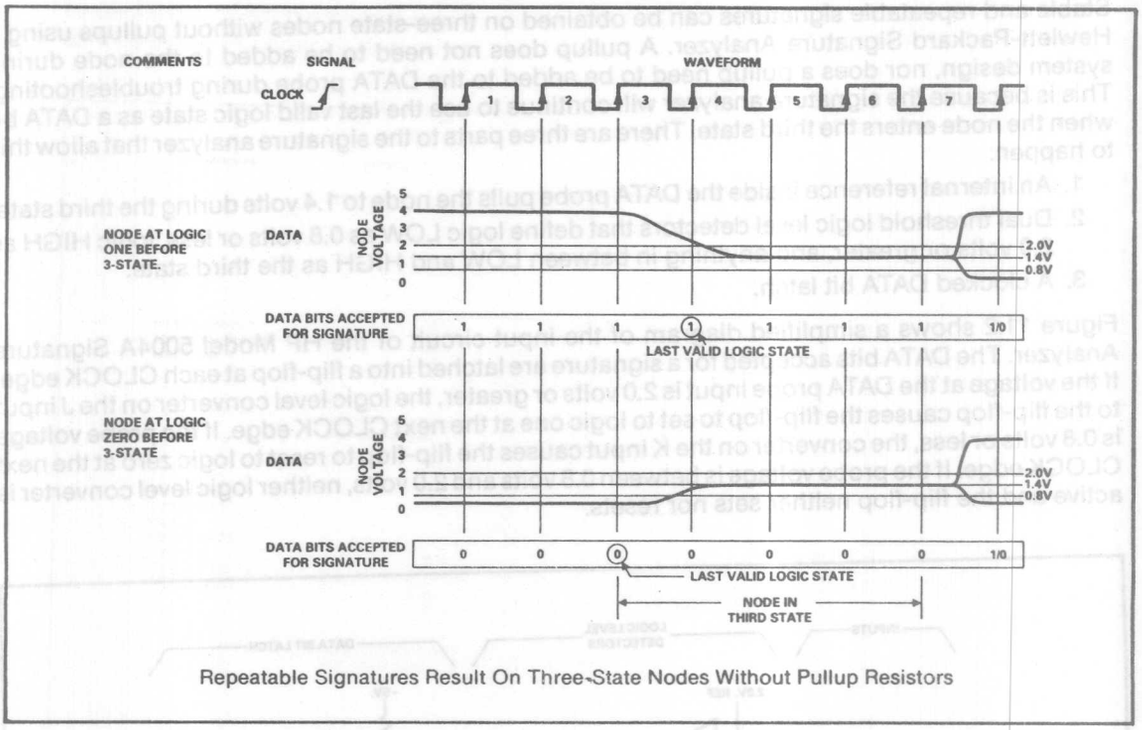
Figure 11.2 shows a simplified diagram of the input circuit of the HP Model 5004A Signature Analyzer. The DATA bits accepted for a signature are latched into a flip-flop at each CLOCK edge. If the voltage at the DATA probe input is 2.0 volts or greater, the logic level converter on the J input to the flip-flop causes the flip-flop to set to logic one at the next CLOCK edge. If the probe voltage is 0.8 volts or less, the converter on the K input causes the flip-flop to reset to logic zero at the next CLOCK edge. If the probe voltage is between 0.8 volts and 2.0 volts, neither logic level converter is active and the flip-flop neither sets nor resets.



Front-End For the HP Model 5004A Signature Analyzer

Figure 11.2

When the node enters the third state, the DATA probe's internal reference pulls the node to 1.4 volts. If the last valid logic state of the node was a high, then the reference pulls the node from logic high to 1.4 volts during the third state and does not allow the node's voltage to drop into the logic zero detection level of 0.8 volts or less. Similarly, a logic low is pulled up to 1.4 volts and a logic high is never detected. This is shown in Figure 11.3. In other words, the last valid logic state remains in the DATA latch and continues to be used for a signature while the node is in the third state. If a node remains in the third state during the entire measurement, then the signature will be all zeroes or the V_{CC} signature depending on whether the last bit on the last node measured was a one or a zero.



Repeatable Signatures Result On Three-State Nodes Without Pullup Resistors

Figure 11.3

Getting V_{CC} Signatures Without Probing V_{CC}

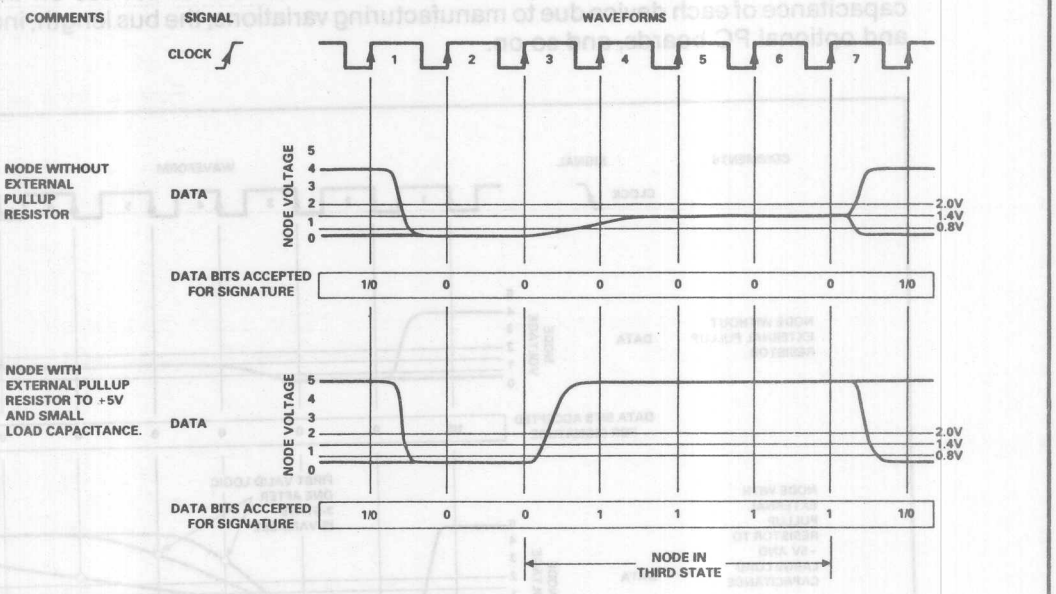
The RESET button on the DATA probe sets the DATA latch to a logic one so that if the probe is left floating (no node is being probed or the node is always in the third state), then all ones are clocked into the signature analyzer as if the probe were placed on +5VDC. This results in a V_{CC} (or +5V, or all ones) signature.

Noise On A Three-State Node

The signature analyzer's treatment of three-state nodes without pullups has been optimized, and proven with several years' experience. The DATA probe input has high enough impedance to reduce circuit loading of the node during the active state and low enough impedance to reduce most noise induced on the node during the third state. This treatment results in stable and repeatable signatures in most cases. However, because of their high impedance, three-state nodes are susceptible to excessive noise induced through crosstalk with high current carrying lines. Although unlikely, excessive noise could cause the logic threshold detectors to switch states exactly at a CLOCK edge and cause an erroneous DATA bit to be sampled resulting in an incorrect, unstable or unrepeatable signature. Section Ten shows how noise on any input affects the signature measurement.

Signatures For Three-State Nodes With Pullups

Stable and repeatable signatures can also be obtained for three-state nodes with pullup resistors in most cases. This is because the pullups on three-state nodes will override the signature analyzer's internal reference and pull the node up to logic high during the third state. This results in logic ones being used as DATA bits during the third state instead of the last valid logic state, as was the case for a node without pullups. See Figure 11.4.



Repeatable Signatures Can Be Obtained On Three-State Nodes With Pullups and Small Load Capacitance

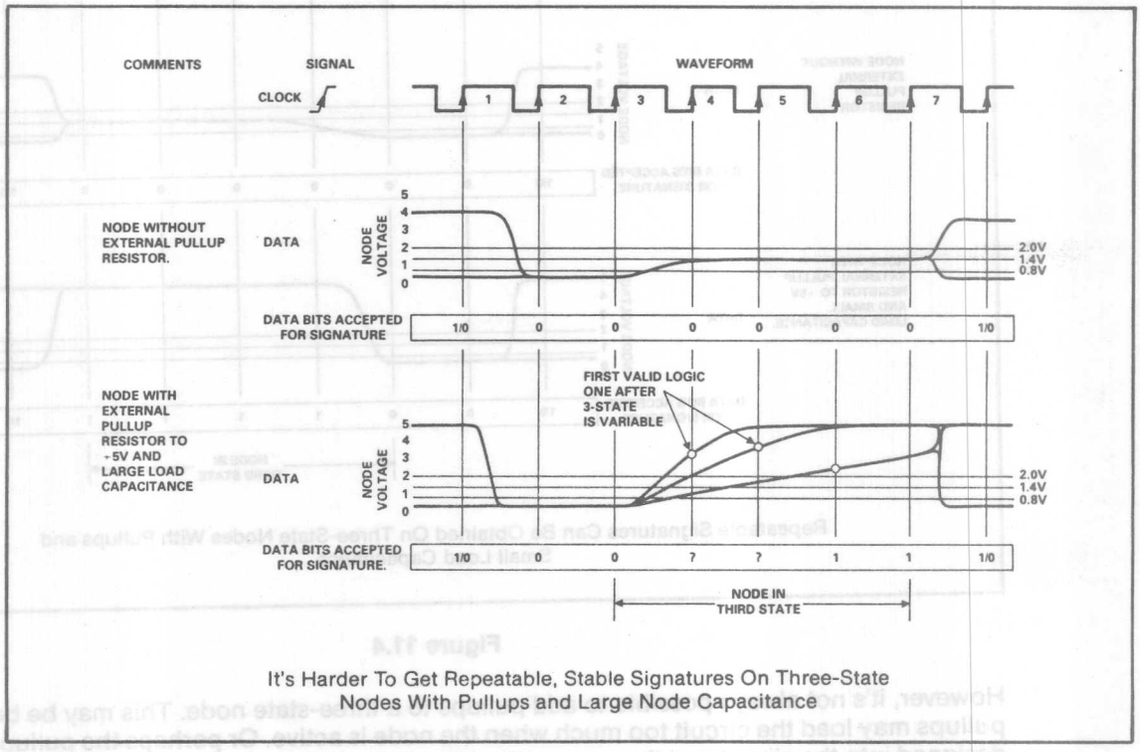
Figure 11.4

However, it's not always possible to add pullups to a three-state node. This may be because the pullups may load the circuit too much when the node is active. Or perhaps the pullups were not designed into the circuit and they cannot be added later. If there are pullups on the node, or if an external pullup is placed on the DATA probe of the signature analyzer, be careful that the signal risetimes remain fast enough.

Nodal Capacitance And Pullups Can Cause Slow Risetimes

Without pullups on the node, it doesn't matter how long it takes for the DATA probe to pull the node to 1.4 volts. Risetimes are not a concern since the DATA bit latch will not change state during the third state. However, external pullups or pulldowns on the node, combined with large nodal capacitance, can cause slow signal risetimes to logic one (or fall times to zero), resulting in erroneous DATA being detected. This can cause incorrect, unstable, or unrepeatable signatures in some cases.

In the case of pullups on the node, the node's voltage will eventually be detected as a logic high during the third state, but the point at which the data will be ones instead of zeroes depends on how fast the node's voltage rises to +5VDC. This is shown in Figure 11.5. The node's RC time constant will determine the rise time. If the risetime is always the same, or if the signal is guaranteed to rise above the logic high threshold before the next CLOCK edge (including time for setup), then a stable signature will result. However, the RC time constant depends on a fixed pullup resistor (if it is designed into the board), or a variable one (if it is placed on the DATA probe during troubleshooting). The RC time constant also depends on the node capacitance that could vary from unit to unit, or even on the same unit over time, or if a part is replaced. The capacitance can also vary, particularly on a large bus, due to the number of devices on the bus, the variable capacitance of each device due to manufacturing variations, the bus length, including backplanes and optional PC boards, and so on.



It's Harder To Get Repeatable, Stable Signatures On Three-State Nodes With Pullups and Large Node Capacitance

Figure 11.5

If unstable or unrepeatable signatures result on a node with a pullup resistor, first verify that a slow risetime is the problem. If it is, then try to find a CLOCK that will sample only valid DATA on the node and will ignore the third state. If this is not possible, then calculate the node's worst case capacitance and choose a pullup resistor that will insure the node's voltage will be above 2.0 volts before the next CLOCK. Be sure all devices on the bus have sufficient drive capability with the new resistor.

If risetime does not seem to be a problem, then check if there is excessive noise synchronous to the CLOCK. If so, eliminate the source of the noise, or choose a CLOCK that will ignore it.

APPENDIX A

Hewlett-Packard Model 5004A Signature Analyzer

Any edge for CLOCK, with any trigger edge selected for START or STOP, in any combination			Min.	Nom.	Max.	Unit	See Fig.
Setup time, t_s	t_{ss}	START	25			ns	A.1
		STOP	25				
	t_{sd}	DATA	15				
Hold time, t_h	t_{hs}	START	0				
		STOP	0				
	t_{hd}	DATA	0				
Clock frequency			0		10	MHz	
Width of CLOCK pulse, t_w high or low			50			ns	
Input impedance to 1.4V		START					
		STOP		50		kΩ	
		CLOCK		7		pf	
		DATA					
V_{ih} High-level input voltage		DATA	1.6	2.0		V	A.2
V_{il} Low-level input voltage				0.8	1.2		
V_{ref} Single logic threshold range		START					A.3
		STOP	0.8	1.4	2.0	V	
		CLOCK					
Hysteresis band around single logic threshold		START					
		STOP		.1		V	
		CLOCK					
Overload protection, all inputs		continuous	−150		150		
		intermittent	−250		+250	V	–
		≤1 min.		250		VAC	

Table A.1—Recommended Operating Conditions

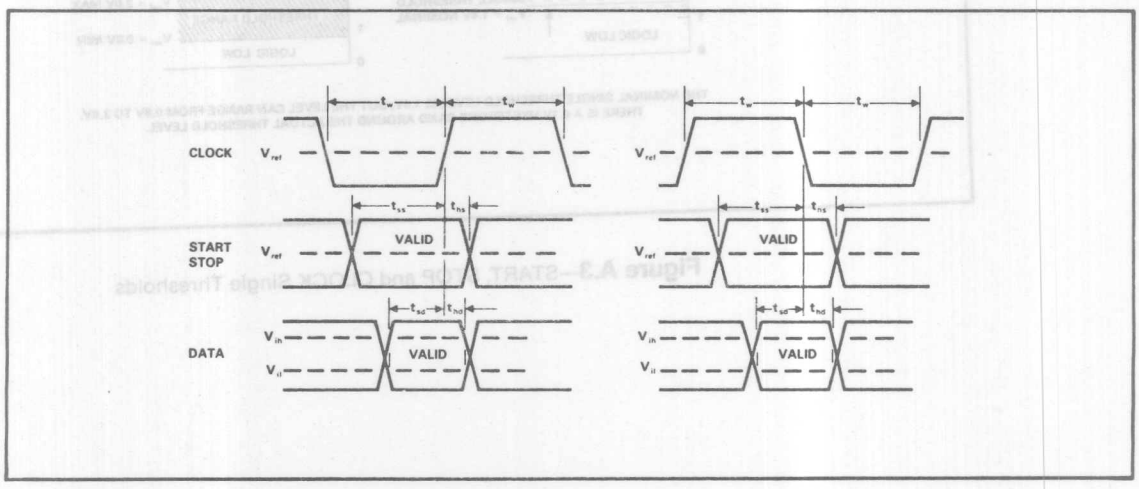


Figure A.1—Voltage Waveforms

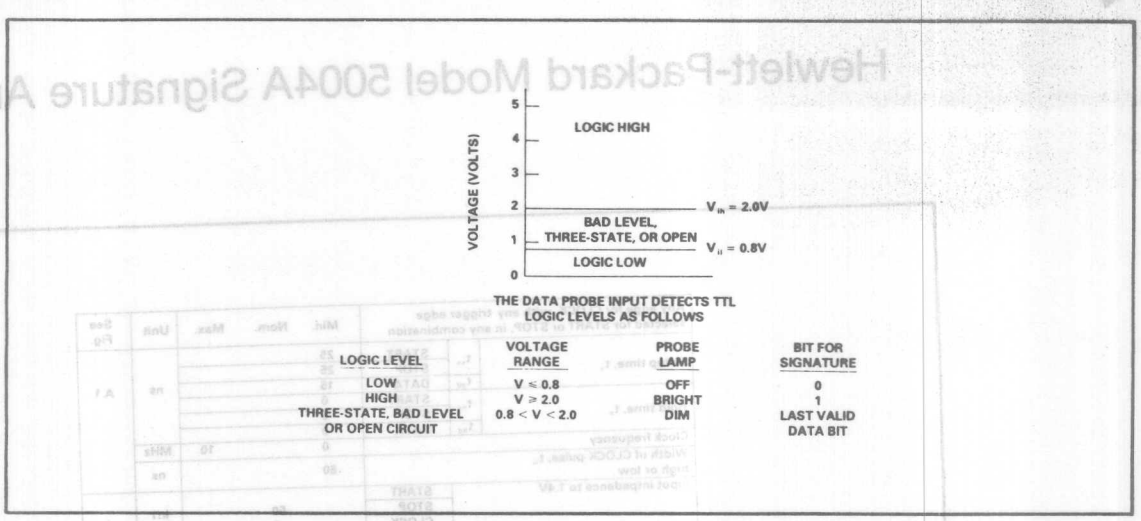


Figure A.2—Data Probe Dual Thresholds (Nominal)

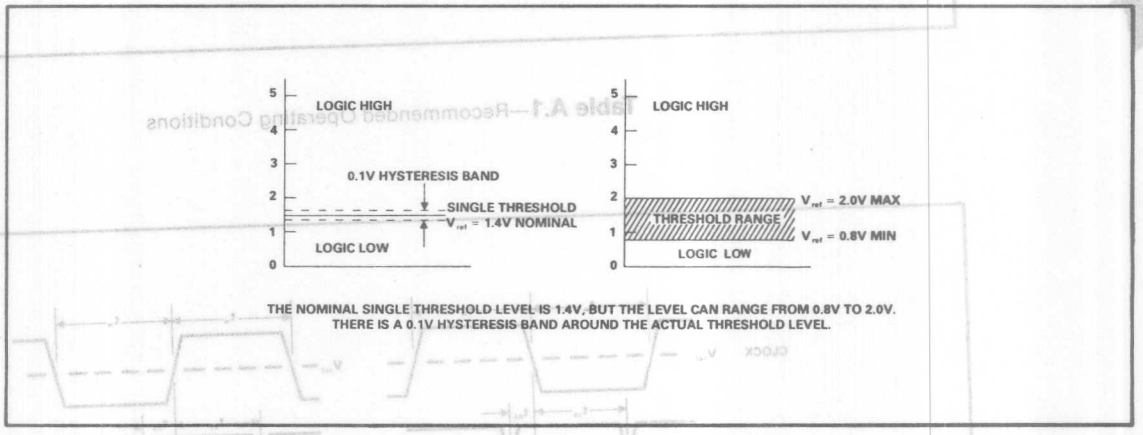
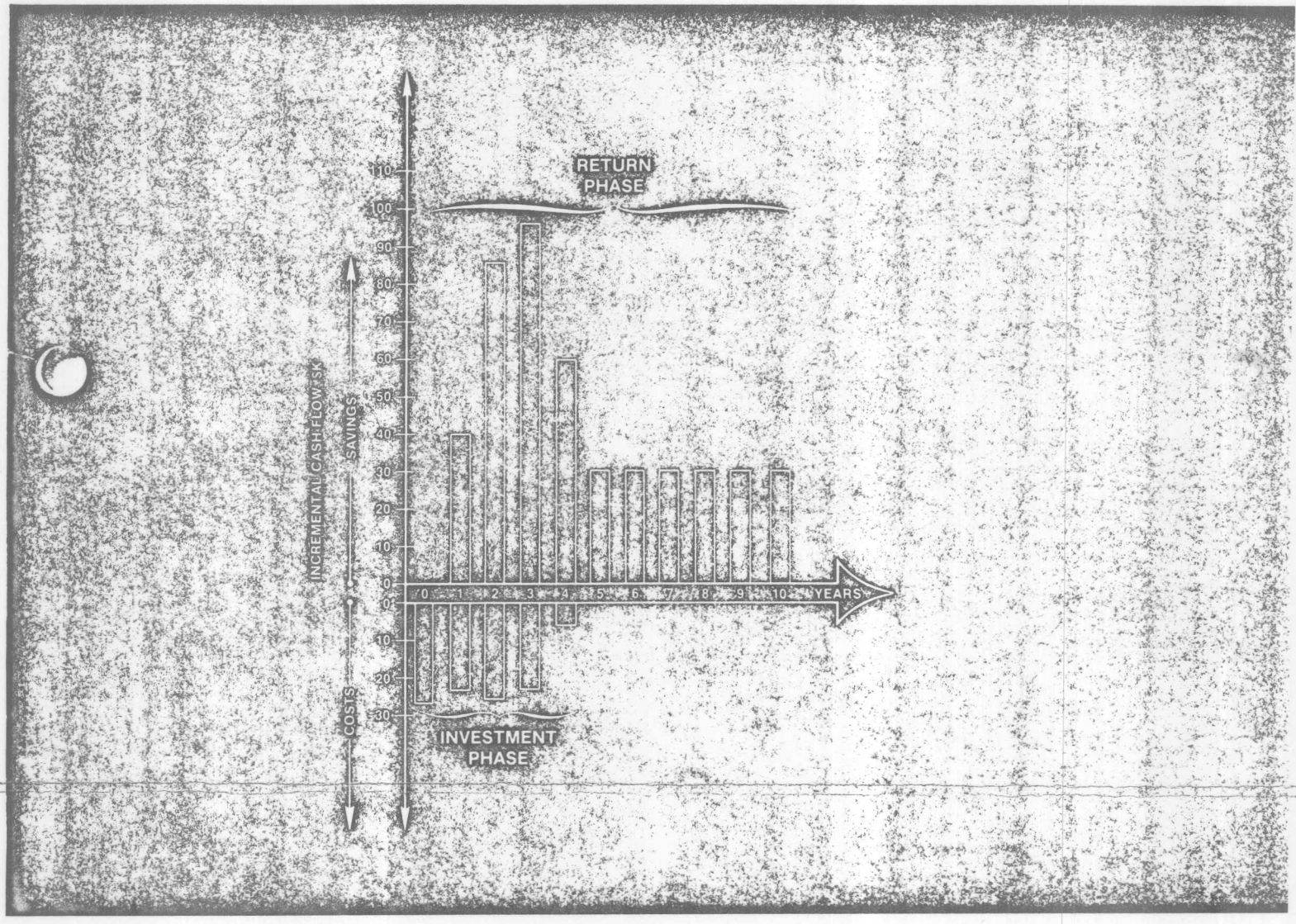


Figure A.3—START, STOP and CLOCK Single Thresholds

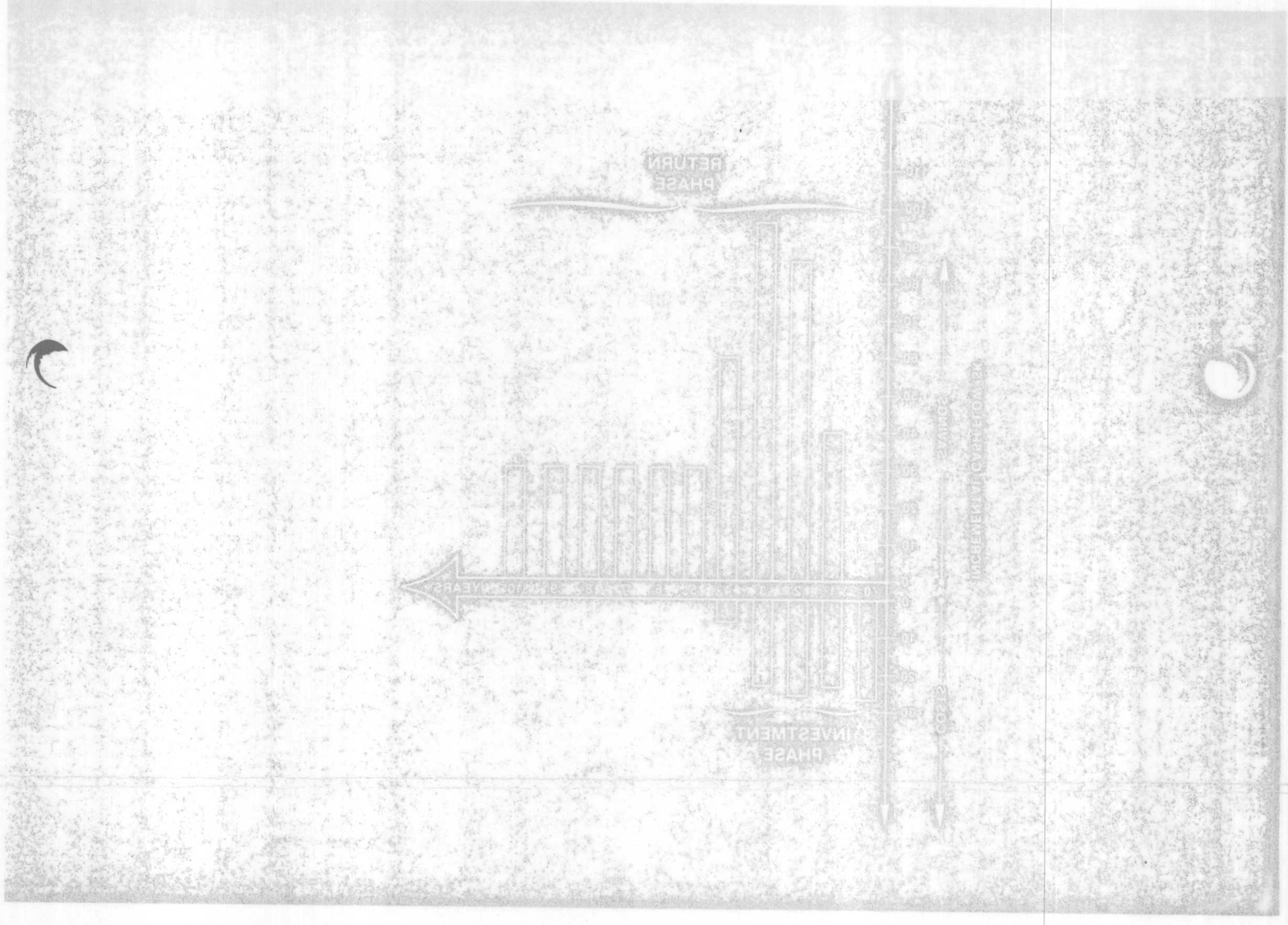
Application Note 222-3
A MANAGER'S GUIDE TO SIGNATURE ANALYSIS



HEWLETT PACKARD

APPLICATION NOTE 222-3
A MANAGER'S GUIDE TO SIGNATURE ANALYSIS

An economic model for determining the cost/feasibility of adopting Signature Analysis in digital test and service.



HEWLETT PACKARD

FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS PUBLICATION

The management decision to adopt S.A. as a digital test/service strategy hinges on this question: Will expected cost reductions in final assembly and field service earn sufficient return on setup investment?

This Application Note advances an economic model for calculating comparative return on investment between different test/service strategies. It suggests simplified rules of thumb for estimating the costs, savings and feasibility of S.A. It concludes with ROI calculations for a sample product. The model is suitable for a wide range of digital products.

ABOUT OTHER PUBLICATIONS

Application Note 222-0, HP publication 02-5952-7593, is a complete index to the latest Signature Analysis Publications.

It lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests from how to design or retrofit Signature Analysis into digital systems to the cost reductions that can be expected in production test and field service by doing so.

It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

CONTENTS

SECTION A—INTRODUCTION

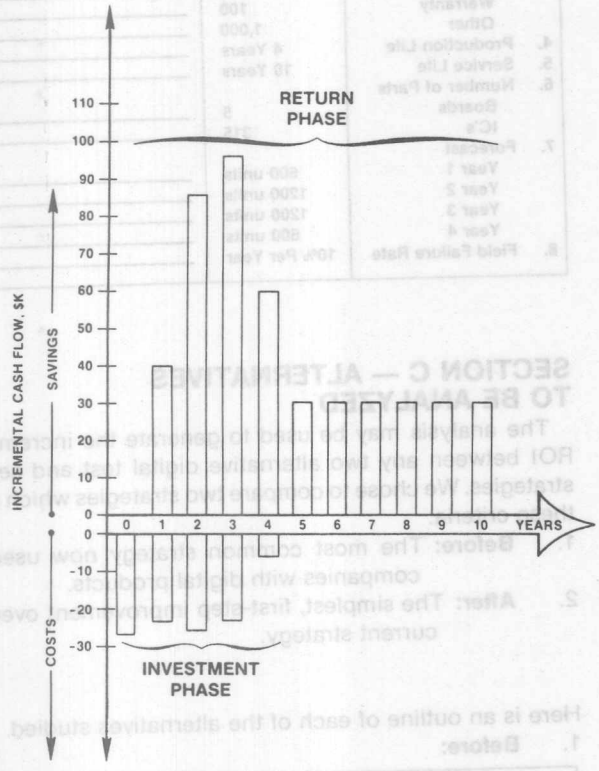
The costs of troubleshooting digital products in final assembly and field service are becoming increasingly visible, and are often perceived to be out of control. Strategies have been advanced which reduce these costs, but which also involve some initial investment in:

- Product Setup
- Test Equipment
- Documentation
- Materials
- Combinations of These Elements

For example, in order to implement Signature Analysis, [1] and take advantage of its savings in labor, processing and test equipment, a product usually needs to be set up, by design or retrofit, to utilize the technique. Therefore, the management decision to adopt or change a test/service strategy for a digital product hinges on the question:

> Will expected cost reductions in final assembly and field service earn sufficient return on the setup investment, and how do the returns for different strategies compare?

1. **Return on Investment.** The comparison of two digital test/service strategies can be considered a return on investment (ROI) exercise. The incremental costs of one strategy over the other are negative cash flows during the *investment* phase of the project. The incremental savings of that strategy are positive cash flows during the *return* phase of the project.



There are several common ROI calculations which allow comparisons of cash flows. This paper utilizes IRR (internal rate of return).

2. **Costs and Savings.** While the ROI calculation is straightforward, the estimation of the cash flows (costs and savings) is not. Existing costs are difficult to measure and proposed savings are difficult to predict. This paper attempts to simplify the exercise by offering some rules of thumb for cost/saving estimation. The rules are very conservative, resulting in higher costs and lower savings than our experience indicates. The effect is a tough comparison, assuring that adoption of a strategy will earn the target ROI.

3. **Model.** The model, then, consists not only of the ROI calculation, but also of a set of simple guidelines for cash flow estimation. It should allow comparison of alternatives in very short study times (one to two days), with minimum research. It is suitable for a wide range of digital products. All calculations are performed on a pocket calculator.

4. **Sample Product.** The paper presents the model via a sample product. The product represents a composite of our experience on hundreds of applications at Hewlett-Packard and other companies.

5. **Organization.** In order to help the reader apply the model to other products, the paper is organized as follows:

Section B — Initial assumptions on the sample product which affect outcome:

- Product Type
- Selling Price
- Cost Structure
- Production Life
- Service Life
- Number of Parts
- Forecast
- Field Failure Rate

Section C — Alternatives to be analyzed and their flow charts:

- Before (or current)
- After (or new)

Section D — Analysis of incremental costs:

- Engineering
- Documentation
- Test Equipment
- Component Stock
- Ongoing Materials

Section E — Analysis of incremental savings:

- Production Labor
- Warranty
- Field Service
- PC Board Stock

Section F — Return on Investment:

- Internal Rate of Return (IRR)
- Partial Implementation, Case 1
- Partial Implementation, Case 2

Section G — Conclusion:

- Summary of Results
- Cross references to tables and figures which allow another product to be studied.

(1) This paper assumes some familiarity with the Signature Analysis technique. For a quick review, check one of the following Hewlett-Packard publications:

(a) Model 5004A Signature Analyzer Data Sheet. (02-5952-7464).

(b) Application Note 222, *A Designer's Guide to Signature Analysis.* (02-5952-7465).

(c) Application Note 222-2, *Application Articles on Signature Analysis.* (02-5952-7542).

SECTION B — INITIAL ASSUMPTIONS

In order to build a realistic economic model, we selected a sample product, and analyzed all of the cost and savings considerations for its digital test and service. So that the process may be applied to other products, this section details the assumptions we made concerning the sample product. It also discusses ways in which these assumptions may affect the results, if varied to accommodate other products. The section concludes with a summary table of the assumptions, with space to state assumptions for another product.

1. **Product Type.** We chose a relatively sophisticated graphic terminal as the sample product, because it incorporates a wide variety of digital troubleshooting challenges: microprocessor, ROM, dynamic RAM, keyboard, CRT controller, character generators, communication ports, etc. The analysis has been applied equally well to both simpler and more complex products.
2. **Selling Price.** The sample product sells for $5,000. The analysis has been used on products ranging from a $300 instrument to a $100,000 ATE system.
3. **Cost Structure.** Any analysis of cost savings depends heavily on the cost breakdown of the product. The larger an existing cost category is, the higher the impact of savings in that category on ROI. For the sample product, we estimated each cost element on the low side, in order to be conservative and lessen the impact of cost savings on ROI.

ELEMENT	COST
Direct Material	$1,000
Direct Labor	$ 250 (25 hours)
Factory Overhead	$ 750
Research and Development	$ 500
Marketing	$ 400
Sales and Service	$ 500
Warranty	$ 100
Other	$1,000

4. **Production Life.** The sample product has an estimated production life of four years. The longer the production life, the greater the impact of cost savings on the ROI model. However, the earlier years have the greatest impact on ROI.
5. **Service Life.** The time during which a product is supported in the field, after production is discontinued, depends on company policy. We assumed a ten-year service life. However, the length of this period has only a minor impact on the ROI calculation.
6. **Number of Parts.** The sample product has 315 IC's, distributed over 5 PC boards. The largest board has 150 IC's, the smallest has 20. There does not appear to be any upper limit on the number of IC's involved. However, it is difficult to show any savings on products of less than 5 IC's.
7. **Sales Forecast.** Assumption of an annual volume forecast over the product life is necessary in order to calculate production, service, and warranty costs and savings. The sample product uses this forecast:

YEAR	QTY SHIPPED	$ VOLUME	SERVICE BASE	WARRANTY BASE
0	0	0	0	0
1	600	$3M	600	600
2	1200	$6M	1800	1200
3	1200	$6M	3000	1200
4	600	$3M	3600	600
5-10	0	0	3600	0

The analysis has been used on products with both higher and lower volumes.

8. **Field Failure Rate.** We assumed a yearly failure rate of 10% of the total installed base for the 10-year service life of the sample product. This is conservatively low, since the higher the failure rate, the greater the savings which can be realized in cutting service costs.

YEAR	FAILURES	IN WARRANTY	OUT OF WARRANTY
0	0	0	0
1	60	60	0
2	180	120	60
3	300	120	180
4	360	60	300
5-10	360	0	360

SUMMARY OF MODEL ASSUMPTIONS

Item	Sample Product	Product Under Study
1. Product Type	Terminal	
2. Selling Price	$5,000	
3. Cost Structure		
Material	$1,000	
Labor	250	
Overhead	750	
R&D	500	
Marketing	400	
Sales/Service	500	
Warranty	100	
Other	1,000	
4. Production Life	4 Years	
5. Service Life	10 Years	
6. Number of Parts		
Boards	5	
IC's	315	
7. Forecast		
Year 1	600 units	
Year 2	1200 units	
Year 3	1200 units	
Year 4	600 units	
8. Field Failure Rate	10% Per Year	

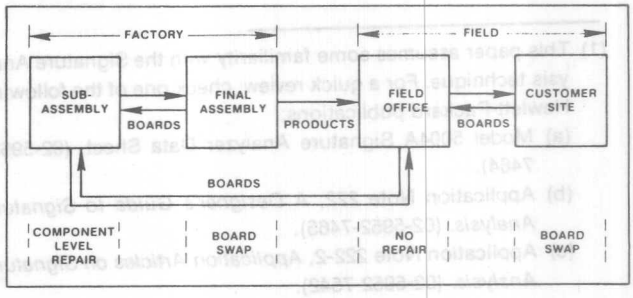
SECTION C — ALTERNATIVES TO BE ANALYZED

The analysis may be used to generate the incremental ROI between any two alternative digital test and service strategies. We chose to compare two strategies which meet these criteria:

1. **Before:** The most common strategy now used by companies with digital products.
2. **After:** The simplest, first-step improvement over the current strategy.

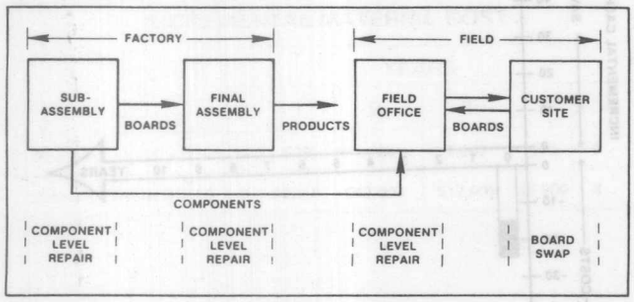
Here is an outline of each of the alternatives studied.

1. **Before:**



This alternative represents the most common strategy for digital test and service. Products which fail in the final assembly area (turn-on, heat-run, final test, QA, etc.) are repaired by swapping PC boards. Bad boards are returned to a subassembly test area for component level troubleshooting and repair. Products which fail in the field are repaired by swapping PC boards at the installation site. Bad boards are returned to the subassembly test area, via the field office, for component level troubleshooting and repair.

2. **After:**

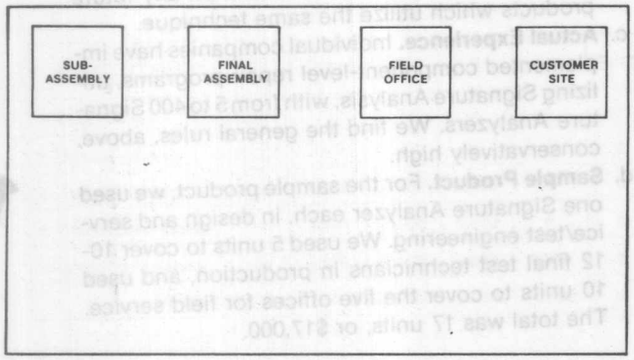


This alternative represents a strategy which could be implemented by setting up the product to be repaired to the component level. Products which fail in the final assembly area are troubleshot and repaired to the component level, without disassembly. Products which fail in the field are repaired by swapping PC boards at the installation site. Bad boards are returned to the field office for component level troubleshooting and repair. No boards return to the factory. We implemented this strategy on the sample product by incorporating the Signature Analysis technique. This required some incremental expenses, over and above what we would have spent to set up the product for straight board-swap repair. However, the savings in repair labor, materials and handling yielded a very attractive ROI. Expenses and savings are detailed in the following sections.

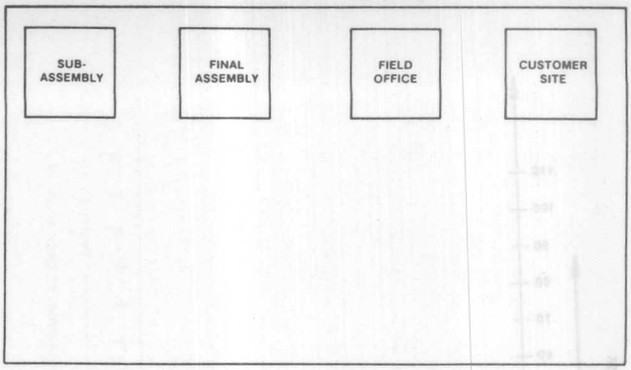
Note: This alternative specified board-swap repair on-site. Additional savings could be generated, with no additional costs, by repairing products on-site, to the component level. There are many examples of this. However, we chose not to take advantage of these savings in the model, since many installation sites are not suitable for replacing components on boards.

3. **Product Under Study.** Here is some space to model comparative test/service strategies for another product.

Before:



After:



Note: Just fill in the appropriate product flows to describe each strategy.

SECTION D — ANALYSIS OF INCREMENTAL COSTS

In order to implement the new strategy for digital test and service in the sample product, we incurred some incremental costs, compared to those for the current strategy:

1. Engineering
2. Documentation
3. Test Equipment
4. Component Stock
5. Ongoing Materials

Here is a detailed analysis of each of the incremental cost areas. The results are summarized at the end of the section, and space is provided to analyze incremental costs for another product.

1. **Engineering**

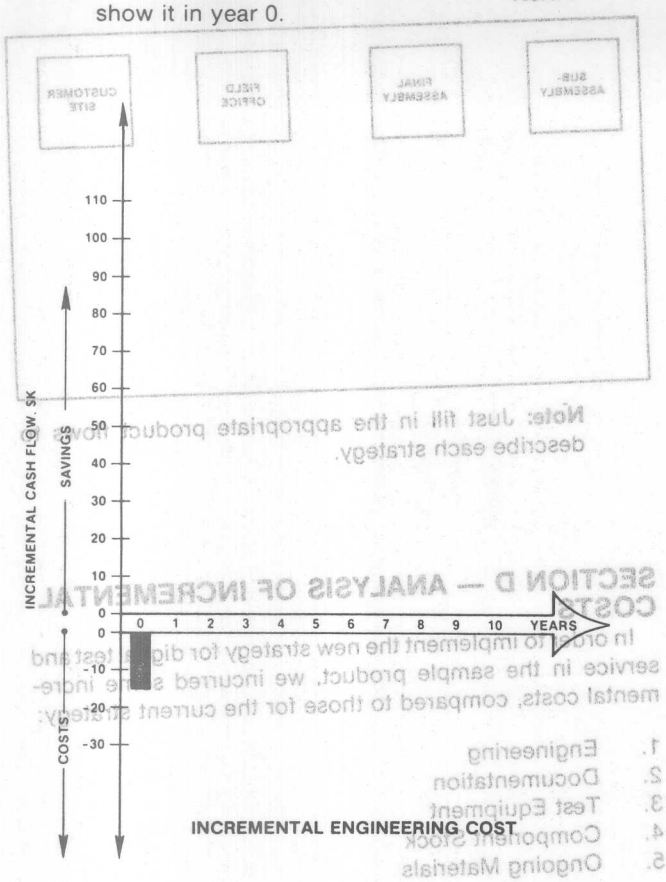
   a. **Description.** In order to set up a product to be repaired to the component level with Signature Analysis, some engineering time is required. The time is devoted to minor hardware re-layout and software modification. If implemented in the design phase, the design team handles it. If implemented as a retrofit, the time generally is spent in the manufacturing engineering area.

   b. **General Rule.** A good, conservative rule is to use 1% incremental engineering time.

   c. **Actual Experience.** We have reported estimates of 1-4 man-weeks of incremental engineering time. This rarely amounts to 1% of the entire design project. The time appears to be the same, whether spent in design or in retrofit. However, there is room for reporting error here, since these are estimates.

   d. **Sample Product.** This product required the equivalent of 5 engineers (all types) for 3 years of design. This is 180 man-months, total. Of this, assume that 2 man-months (1%) were devoted to setting up the product for component-level repair. We used a conservatively high figure of $7,500 per month for fully loaded design time, for a total incremental cost of $15,000.

e. **Timing.** The engineering time is spent at the end of the design cycle, just before production. We show it in year 0.



INCREMENTAL ENGINEERING COST

2. **Documentation**

a. **Description.** Troubleshooting procedures are more thorough for component level repair, than for board-swap. Signature Analysis greatly reduces the burden; however, there is some incremental time involved:

- gathering signatures.
- writing procedures.
- verifying both.

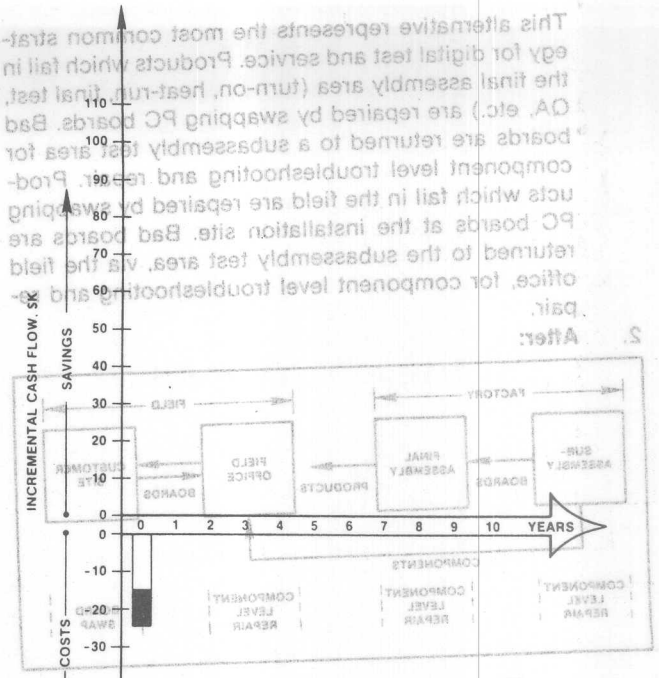
The time is spent in product support engineering, manufacturing test engineering or service engineering.

b. **General Rule.** A good, conservative guideline is to use 2 man-months for incremental documentation time.

c. **Actual Experience.** We have reported estimates of 2-4 man-weeks of incremental documentation time, so the general rule is quite conservative. Remember that this is only the incremental time spent to add Signature Analysis instructions to the procedures which would have been prepared under the current strategy.

d. **Sample Product.** Two man-months of documentation effort were targeted at a loaded cost of $5,000 per month, for a total incremental cost of $10,000.

e. **Timing.** The documentation time is spent in the pre-production and early production life of the product. We show it in year 0.



INCREMENTAL DOCUMENTATION COST

3. **Test Equipment**

a. **Description.** Utilizing the Signature Analysis technique requires purchase of Signature Analyzers, at $1,000 each, in these locations:

(1) Lab — for aiding in the design or retrofit of Signature Analysis into the product.

(2) Service or Test Engineering — for documenting the troubleshooting procedure.

(3) Production — for troubleshooting in the final assembly area.

(4) Field Offices — for field service troubleshooting of returned boards.

b. **General Rule.** Plan on one Signature Analyzer for the lab for design, and one for service/test engineering for documentation. In the final assembly area, plan on one unit per test position or, if the unit can be shared, one per 3 final test technicians. In the field, plan on one unit per office, initially. (The upper limit for field service will vary. One guideline is to put one unit in the field for each field service technician. Another guideline would be to put one unit into each field office for each 50 projected repairs per year.) Each Signature Analyzer, of course, is usable on any future products which utilize the same technique.

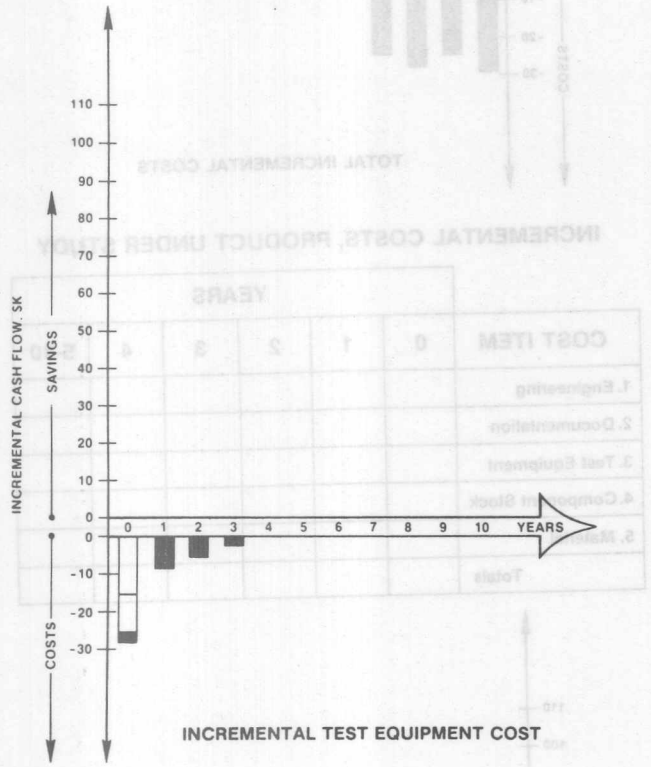
c. **Actual Experience.** Individual companies have implemented component-level repair programs, utilizing Signature Analysis, with from 5 to 400 Signature Analyzers. We find the general rules, above, conservatively high.

d. **Sample Product.** For the sample product, we used one Signature Analyzer each, in design and service/test engineering. We used 5 units to cover 10-12 final test technicians in production, and used 10 units to cover the five offices for field service. The total was 17 units, or $17,000.

e. **Timing.** These 17 Signature Analyzers were acquired over 4 years, as shown.

INCREMENTAL SIGNATURE ANALYZER COST

		YEARS					
LOCATION	UNITS	0	1	2	3	4	5-10
Lab	1	1	0	0	0	0	0
Service/Test Engineering	1	1	0	0	0	0	0
Production	5	0	3	2	0	0	0
Field Service	10	0	5	3	2	0	0
Total Units	17	2	8	5	2	0	0
Cost @ $1,000	$17,000	$2,000	$8,000	$5,000	$2,000	0	0



INCREMENTAL TEST EQUIPMENT COST

4. **Component Stock**

a. **Description.** Since the new strategy calls for component-level repair in field offices, we require parts kits, which would not have been stocked currently. These are startup parts kits, which are added during the life of the product, as required by the growing installed base. We do not show factory parts stock here, since it is approximately the same under either strategy.

b. **General Rule.** Itemize one of each IC per kit. Plan on enough kits to handle 2 months' failures, at the predicted mature failure rate. This is very conservative, since successive failures rarely require the same part at the same location.

c. **Actual Experience.** Most service groups have found that they do not require all electronic parts to be in a kit, and that they can understock multiple usage parts. Actual stocking requirements come in well under the general rule.

d. **Sample Product.** The sample product has 315 IC's (Section B-6), the costs of which are:

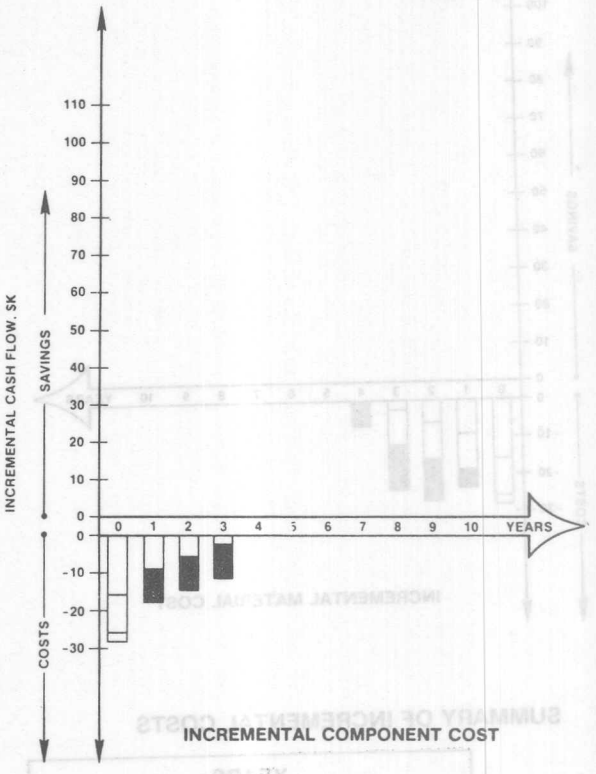
300 IC's at $1.00 each	=	$300.00
15 IC's at $10.00 each	=	$150.00
Total cost per kit		$450.00

The mature failure rate was projected at 30 per month (Section B-8), so two months' failures are covered by 60 kits, or $27,000.

e. **Timing.** We acquire the 60 kits over a period of 3 years, keeping well ahead of the projected installed base, as follows:

INCREMENTAL COMPONENT COST

		YEARS					
	QTY REQUIRED	0	1	2	3	4	5-10
Field Parts Kits	60	0	20	20	20	0	0
Kit Costs at $450.00	$27,000	0	$9,000	$9,000	$9,000	0	0



INCREMENTAL COMPONENT COST

5. **Ongoing Materials**

a. **Description.** When setting up a product for Signature Analysis troubleshooting, there are generally some additional parts required in the product itself. These usually consist of switches, jumpers, test points, a socket, etc. (Active components are rarely required.) These parts then contribute to a small incremental material cost.

b. **General Rule.** Add 1% of the standard material cost.

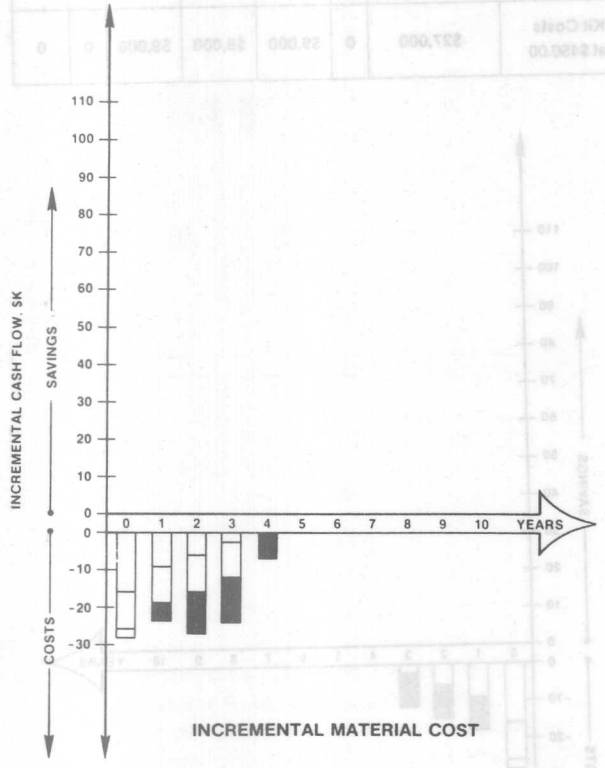
c. **Actual Experience.** Incremental material costs (assuming no additional ROM space is required) range from 0–$5.00 per unit.

d. **Sample Product.** Using the 1% rule on the $1,000 material cost (Section B-3), we have a conservatively high incremental material cost of $10.00 per unit. This is then applied to the forecast (Section B-7) to obtain the annual incremental cost.

e. **Timing.** Material costs are incurred as follows:

INCREMENTAL MATERIAL COST

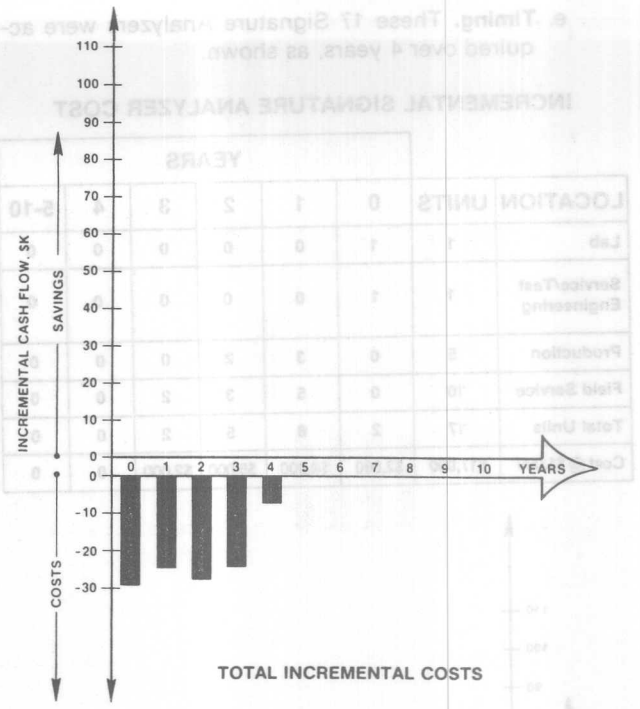
	YEARS					
	0	1	2	3	4	5-10
Forecast, Units	0	600	1,200	1,200	600	0
Annual Cost at $10.00	0	$6,000	$12,000	$12,000	$6,000	0



INCREMENTAL MATERIAL COST

SUMMARY OF INCREMENTAL COSTS

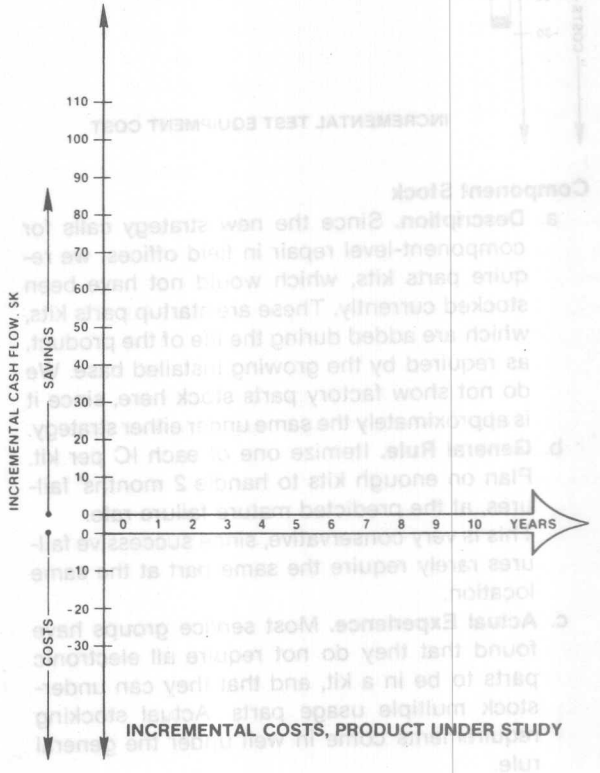
	YEARS					
COST ITEM	0	1	2	3	4	5-10
1. Engineering	15,000	0	0	0	0	0
2. Documentation	10,000	0	0	0	0	0
3. Test Equipment	2,000	8,000	5,000	2,000	0	0
4. Component Stock	0	9,000	9,000	9,000	0	0
5. Ongoing Materials	0	6,000	12,000	12,000	6,000	0
Totals	$27,000	$23,000	$26,000	$23,000	$6,000	0



TOTAL INCREMENTAL COSTS

INCREMENTAL COSTS, PRODUCT UNDER STUDY

	YEARS					
COST ITEM	0	1	2	3	4	5-10
1. Engineering						
2. Documentation						
3. Test Equipment						
4. Component Stock						
5. Material						
Totals						



INCREMENTAL COSTS, PRODUCT UNDER STUDY

SECTION E — ANALYSIS OF INCREMENTAL SAVINGS

By implementing the new digital test/service strategy, we experience significant incremental savings over the current strategy, in these cost areas:

1. Production Labor
2. Warranty
3. Field Service
4. PC Board Stock

Here is a detailed analysis of each of the incremental savings areas. The results are summarized at the end of the section, and space is provided to analyze incremental savings for another product.

1. **Production Labor**

   a. **Description.** The new strategy utilizes Signature Analysis to generate substantial savings in the final assembly troubleshooting area. This is because a final test technician can now make component level repairs in about the same time as it formerly took just to locate and verify a bad board.

   b. **General Rule.** Isolate the average time per unit normally spent on troubleshooting and repairing an assembled product on the line, and reduce it, 2:1.

   c. **Actual Experience.** Most cases do not generate "before/after" data since, when Signature Analysis is employed, the "before" case is never practiced. However, those cases we have studied show time improvements from 4:1 to 8:1. Therefore, the general rule, above, is very conservative.

   d. **Sample Product.** Using the 2:1 rule, and a cost breakdown for the sample product (Section B-3), we calculate production labor savings of $50.00 per unit.

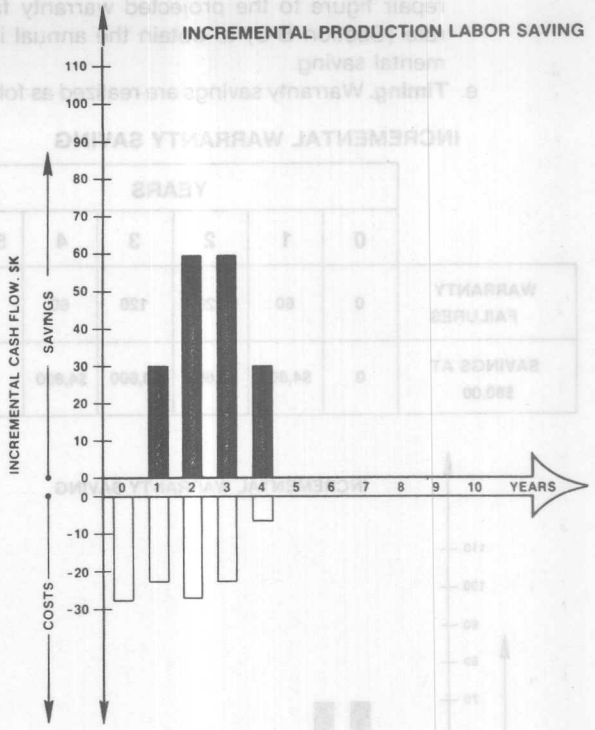
	HOURS PER UNIT		
TYPE OF LABOR	BEFORE	AFTER	SAVED
SUBASSEMBLY	4	4	0
FINAL ASSEMBLY	6	6	0
TEST	5	5	0
TROUBLESHOOTING	10	5	5
TOTAL HOURS	25	20	5
COST AT $10.00 PER HOUR	$250.00	$200.00	$50.00

This unit saving is then applied to the forecast (Section B-7) to obtain the annual incremental saving.

   e. **Timing.** Production labor savings are realized as follows:

INCREMENTAL PRODUCTION LABOR SAVING

	YEARS					
	0	1	2	3	4	5-10
FORECAST	0	600	1,200	1,200	600	0
ANNUAL SAVINGS AT $50.00	0	$30,000	$60,000	$60,000	$30,000	0



2. **Warranty**

   a. **Description.** By repairing PC boards in the field, instead of at the factory, we generate savings in the operation of a board exchange program in the areas of:
   - inventory.
   - administration.
   - logistics/distribution.
   - no-trouble-found boards.

   The repair cost reduction impacts both warranty and field service savings.

   b. **General Rule.** This can be a complex area in which to make estimates. However, the following formula is conservatively safe:

   Savings per repair =
   Average board exchange price (before).
   Less: Average repair parts price (after).
   Less: Incremental field repair labor (after).

   Some rules of thumb are:
   (1) Average board exchange price - $100.00.
   (2) Average repair parts price - $10.00.
   (3) Average incremental field repair labor - 1 hour = $10.00.

   Therefore, a conservatively low saving figure would be $80.00 per repair. This assumes that the former board exchange fee was set in such a way as to just cover the costs of running the program.

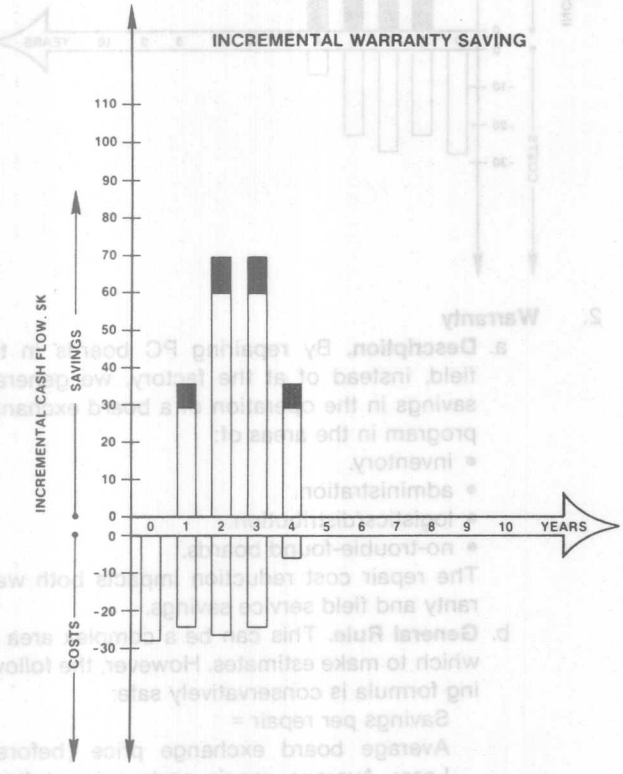
   c. **Actual Experience.**
   (1) Board exchange prices range from $50.00 to $500.00.
   (2) Repair component prices range from $.50 to $50.00.
   (3) Incremental labor is often well below the minimum charge, since boards can usually be repaired at the field office in the same time it formerly took to verify the bad board, process paper work and handle shipping.
   (4) The $80.00 figure is conservative.

d. **Sample Product.** We applied the $80.00-per-repair figure to the projected warranty failure rate (Section B-8) to obtain the annual incremental saving.

e. **Timing.** Warranty savings are realized as follows:

INCREMENTAL WARRANTY SAVING

	YEARS					
	0	1	2	3	4	5-10
WARRANTY FAILURES	0	60	120	120	60	0
SAVINGS AT $80.00	0	$4,800	$9,600	$9,600	$4,800	0



3. **Field Service**

a. **Description.** Same as 2-a, above.

b. **General Rule.** $80.00 per repair.

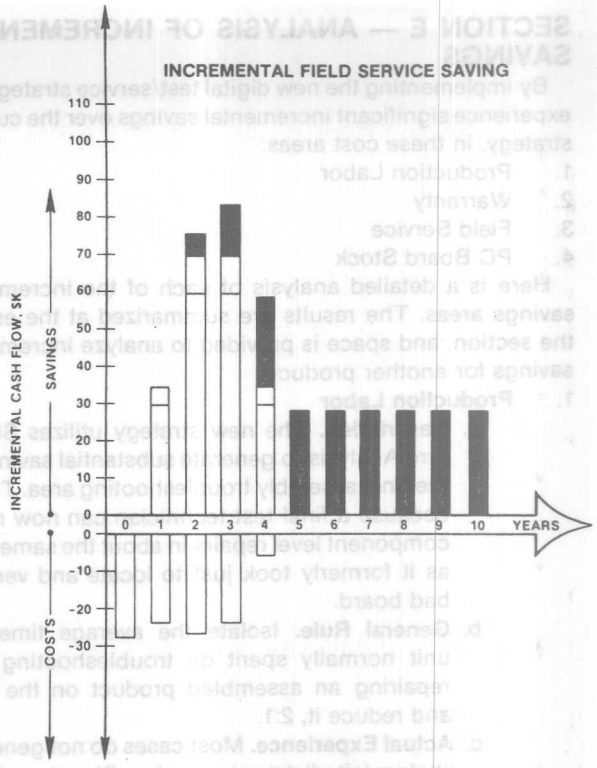
c. **Actual Experience.** Same as 2-c, above.

d. **Sample Product.** We applied the $80.00-per-repair figure to the projected non-warranty failure rate (Section B-8) to obtain the annual incremental saving.

e. **Timing.** Field service savings are realized as follows:

INCREMENTAL FIELD SERVICE SAVING

	YEARS					
	0	1	2	3	4	5-10
NON—WARRANTY FAILURES	0	0	60	180	300	360
SAVINGS AT $80.00	0	0	$4,800	$14,400	$24,000	$28,800



4. **PC Board Stock**

a. **Description.** The new strategy requires parts kits in field offices for component-level repair (Section D-4). Because of that, fewer board kits will be required in field offices. Board kits will be required only for on-site repair, with the bad boards being repaired at the field office. The reduction in startup board kits constitutes an incremental saving for the new strategy.

b. **General Rule.** Calculate the number of board kits required under the current strategy, enough to cover 2 months' failures at the projected mature failure rate. Calculate the number of board kits required under the new strategy, about one-third of the current figure. The difference in numbers of kits is the saving. A good approximation of kit cost is the product's material cost. Usually, with Signature Analysis, a stripped mainframe of the product is required in each office, in order to power the board for troubleshooting. So, from the board kit saving, be sure to deduct the cost of a mainframe for each office. Again, the product material cost should cover this.

c. **Actual Experience.** Reports indicate that the reduction of board kits is usually more than outlined above. The current strategy often requires 3 months' failure coverage, due to pipeline and turnaround problems. The new strategy often requires only 1 board kit per field office.

d. **Sample Product.**

(1) **"Before"** — Projected mature failure rate is 30 per month (Section B-8). Two months' coverage would be 60 kits. Cost per kit (material cost in Section B-3) is $1,000, for a total of $60,000.

(2) **"After"** — One-third of the "before" figure is 20 kits, or $20,000.

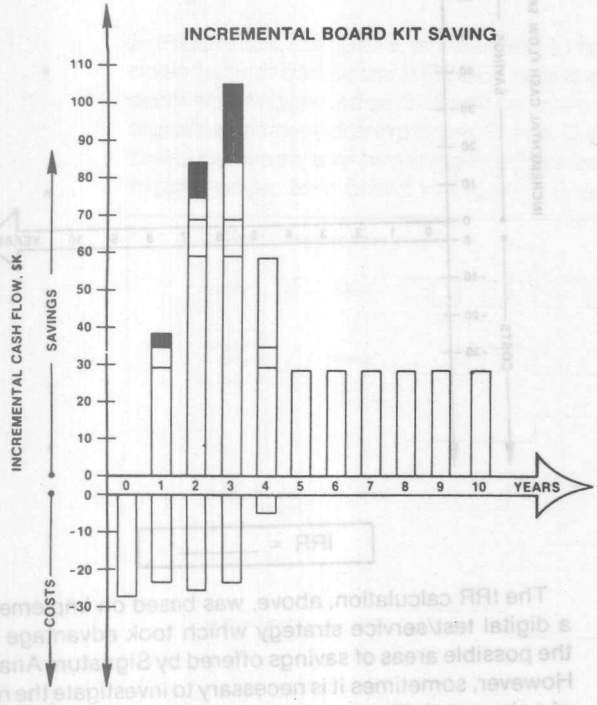
(3) **Mainframes** — One stripped mainframe, at material cost, for each of the five field offices amounts to $5,000.

(4) **Total Saving** — $60,000 less $20,000, less $5,000 = $35,000.

e. **Timing.** The $35,000 saving in board kits is distributed over 3 years, as the kits would have flowed into the field pipeline with the growing service base.

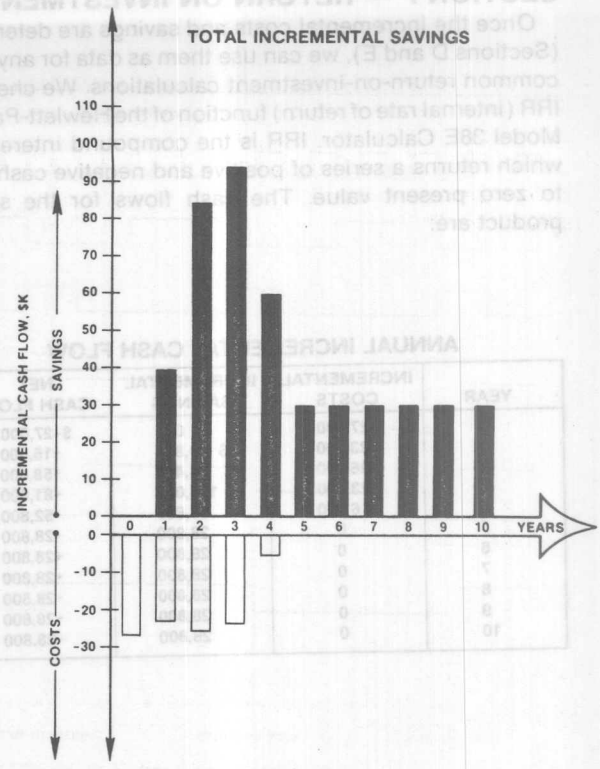
INCREMENTAL BOARD KIT SAVING

	YEARS					
	0	1	2	3	4	5-10
KITS REQUIRED, BEFORE	0	20	20	20	0	0
LESS KITS REQUIRED, AFTER	0	−10	−10	0	0	0
KITS SAVED	0	10	10	20	0	0
COST SAVINGS AT $1,000	0	$10,000	$10,000	$20,000	0	0
LESS SERVICE MAIN-FRAMES, 5 AT $1,000	0	−$ 5,000	0	0	0	0
NET SAVINGS	0	$ 5,000	$10,000	$20,000	0	0



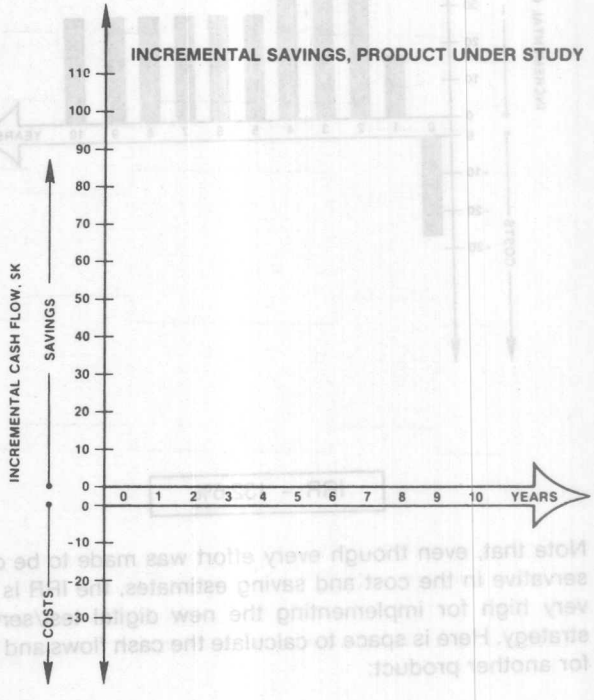
SUMMARY OF INCREMENTAL SAVINGS

	YEARS					
SAVINGS ITEMS	0	1	2	3	4	5-10
1. Production Labor	0	30,000	60,000	60,000	30,000	0
2. Warranty	0	4,800	9,600	9,600	4,800	0
3. Field Service	0	0	4,800	14,400	24,000	28,800
4. PC Board Stock	0	5,000	10,000	20,000	0	0
Totals	0	$39,800	$84,400	$104,000	$58,800	$28,800



INCREMENTAL SAVINGS, PRODUCT UNDER STUDY

	YEARS					
SAVINGS ITEMS	0	1	2	3	4	5-10
1. Production Labor						
2. Warranty						
3. Field Service						
4. PC Board Stock						
Totals						

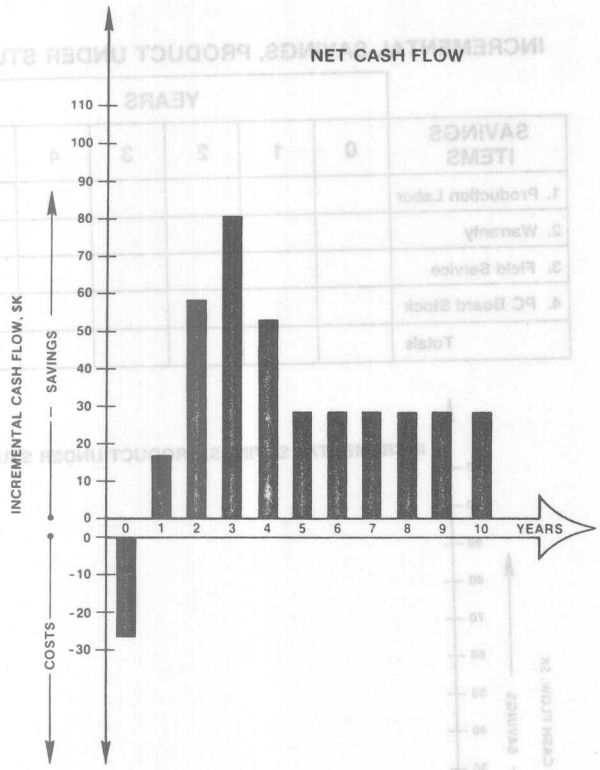


SECTION F — RETURN ON INVESTMENT

Once the incremental costs and savings are determined (Sections D and E), we can use them as data for any of the common return-on-investment calculations. We chose the IRR (internal rate of return) function of the Hewlett-Packard Model 38E Calculator. IRR is the compound interest rate which returns a series of positive and negative cash flows to zero present value. The cash flows for the sample product are:

ANNUAL INCREMENTAL CASH FLOW

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	23,000	$ 39,800	+16,800
2	26,000	84,400	+58,400
3	23,000	104,000	+81,000
4	6,000	58,800	+52,800
5	0	28,800	+28,800
6	0	28,800	+28,800
7	0	28,800	+28,800
8	0	28,800	+28,800
9	0	28,800	+28,800
10	0	28,800	+28,800

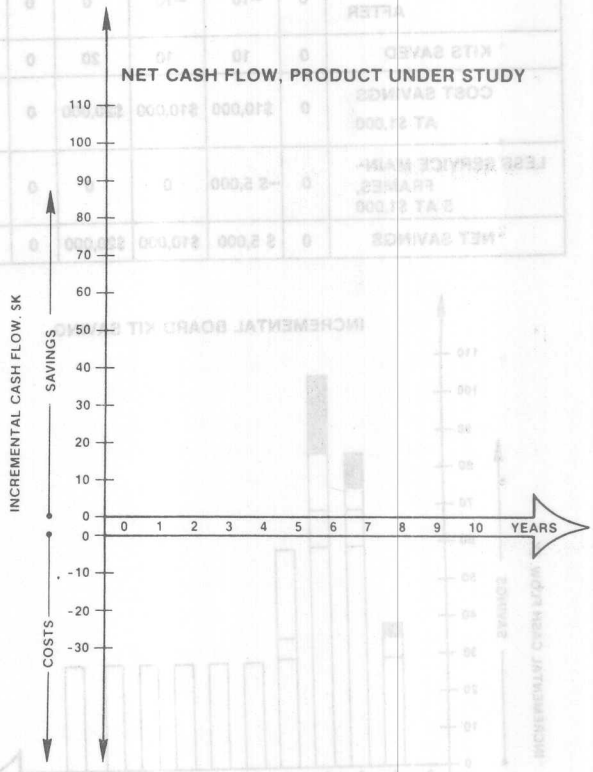


IRR = 132.6%

Note that, even though every effort was made to be conservative in the cost and saving estimates, the IRR is still very high for implementing the new digital test/service strategy. Here is space to calculate the cash flows and IRR for another product:

INCREMENTAL CASH FLOW, PRODUCT UNDER STUDY

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			



IRR = _____

The IRR calculation, above, was based on implementing a digital test/service strategy which took advantage of all the possible areas of savings offered by Signature Analysis. However, sometimes it is necessary to investigate the return of only a partial implementation of the technique. Here are two cases of interest:

Case 1 — Include only production and warranty savings. Do not include field service savings, since the savings here may be passed on to customers via lower service charges. Or, the savings may be realized by a third-party service organization, not by the manufacturer.

Case 2 — Include only production savings. This case assumes that Signature Analysis is used only in final assembly test, not in field service at all.

Case 1—All incremental costs are included (Section D) but only the following incremental savings are included:

- Production Labor (Section E-1)
- Warranty (Section E-2)
- PC Board Stock (Section E-4)

Field service savings are not included.

Case 2—Only those incremental costs are included which are required to set up a product for Signature Analysis in final assembly test:

- Engineering (Section D-1)
- Documentation (Section D-2)
- Test Equipment (Section D-3), reduced from 17 units to 7.
- Ongoing Materials (Section D-5).

Only those incremental savings are included which are realized in final assembly test:

- Production Labor (Section E-1).

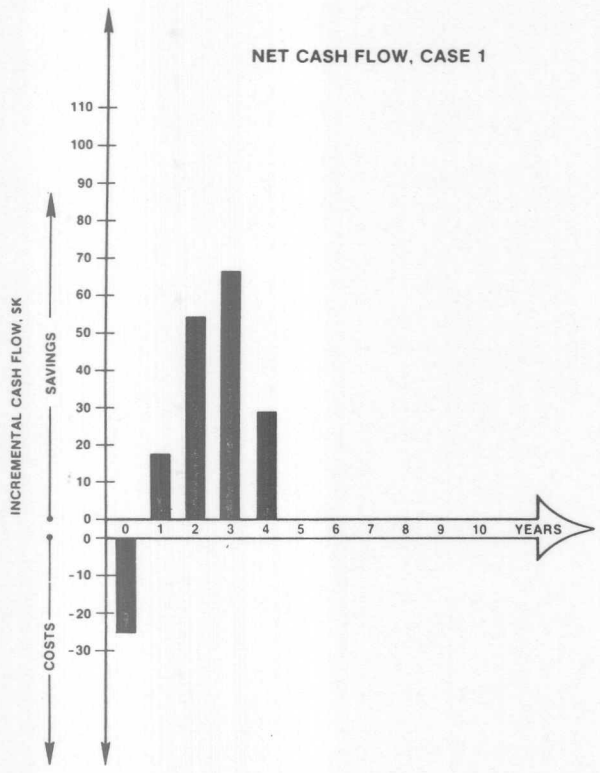
Warranty, field service and PC board stock savings are not included.

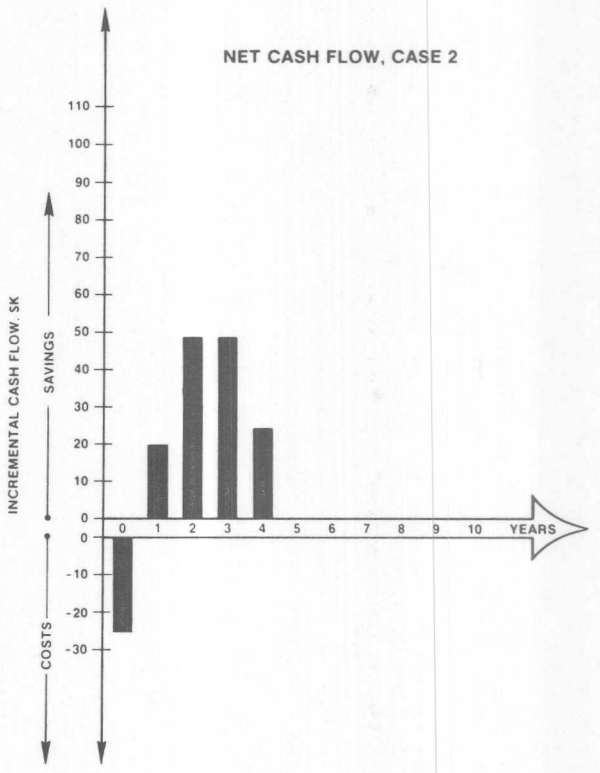
ANNUAL INCREMENTAL CASH FLOW, CASE 1

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	23,000	$39,800	+16,800
2	26,000	79,600	+53,600
3	23,000	89,600	+66,600
4	6,000	34,800	+28,800
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

ANNUAL INCREMENTAL CASH FLOW, CASE 2

YEAR	INCREMENTAL COSTS	INCREMENTAL SAVINGS	NET CASH FLOW
0	$27,000	0	$-27,000
1	11,000	$30,000	+19,000
2	12,000	60,000	+48,000
3	12,000	60,000	+48,000
4	6,000	30,000	+24,000
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0





IRR = 116.8%

In this case, eliminating field service savings still results in a very attractive IRR.

IRR = 107.4%

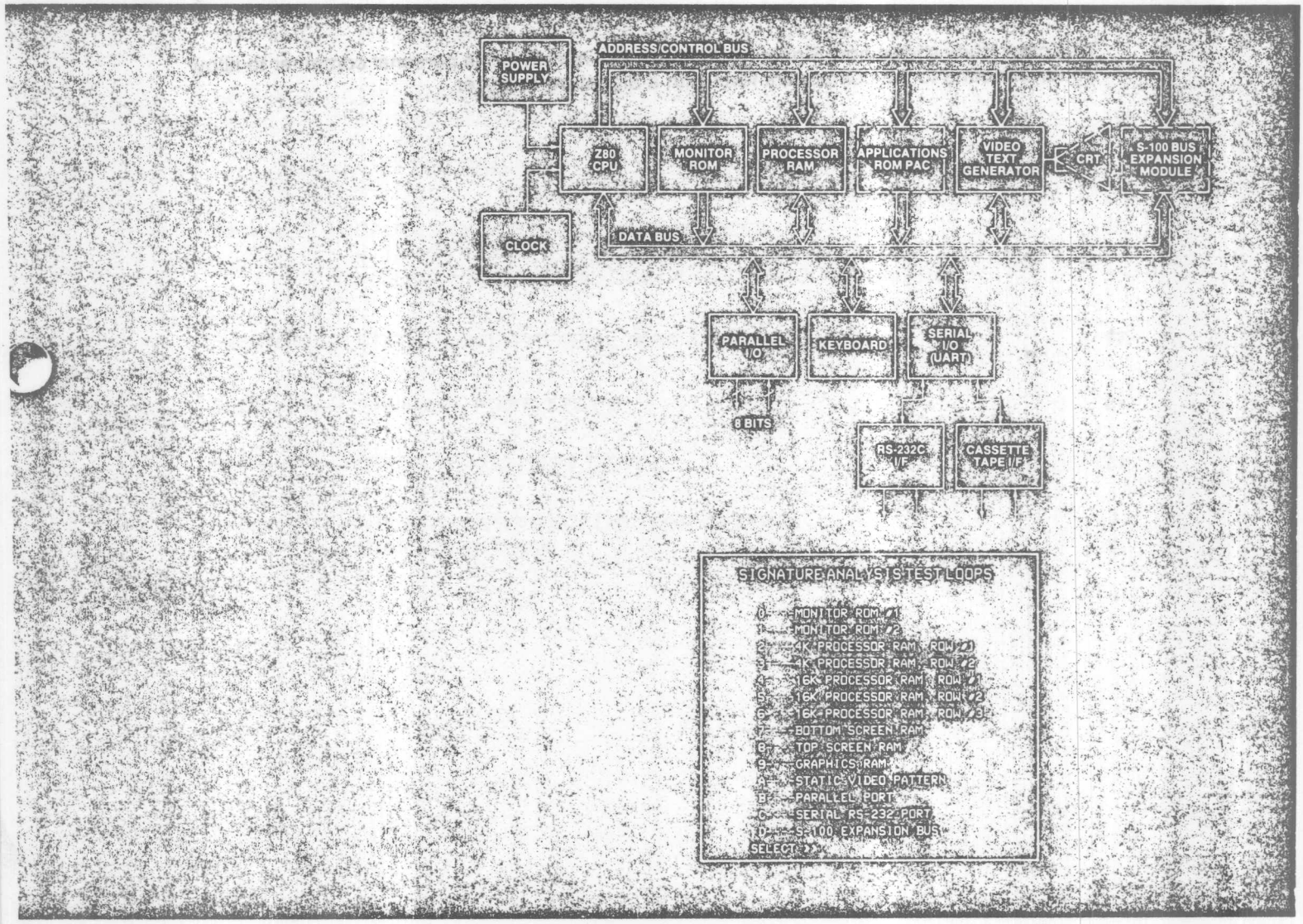
Even in this case, IRR is over 100%, in a situation where 20-30% is considered attractive.

SECTION G — CONCLUSION

1. **Summary.** The use of the model to compare two digital test/service strategies can also show the IRR derived by switching from one (current) strategy to another (new) one. A significant finding was that, even when the new Signature Analysis strategy was only partially implemented, the IRR was over 100%. This can be important in getting a new strategy started, by reducing the number of departments which must collaborate. In Sample Case 2, the Production Department could have justified the project on production savings alone. The Service Department would have had the option of adopting it later.

2. **Next Step.** The reader may apply the model to another product by utilizing the blank tables and plots in the paper. Here is an index to them:

Application Note 222-10
A SIGNATURE ANALYSIS CASE STUDY
of a Z80-Based Personal Computer

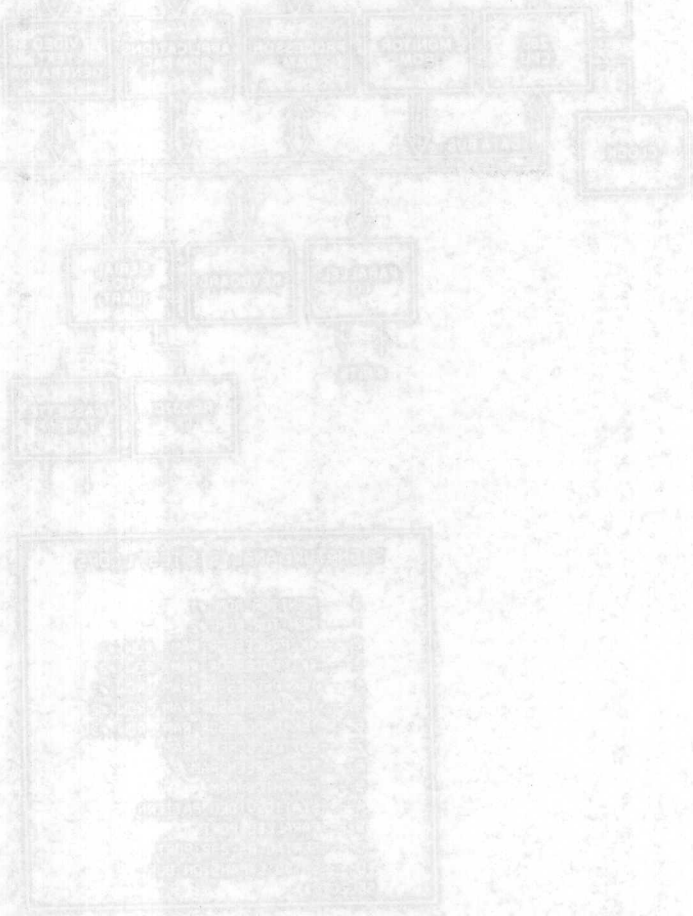


HEWLETT PACKARD

APPLICATION NOTE 222-10

A SIGNATURE ANALYSIS CASE STUDY of a Z80-Based Personal Computer

This case study shows how Signature Analysis was retrofit into a personal computer to allow troubleshooting to the component level with a Signature Analyzer.



FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS CASE STUDY SERIES

Use of a Signature Analyzer requires that some test features be designed or retrofit into the product to be tested. This application note is one in a series of case studies aimed at assisting designers, test engineers, and others in understanding these features so that they can easily add Signature Analysis to their product. These case studies show detailed examples of these features in various digital systems based on specific microprocessors.

ABOUT THIS PUBLICATION

This note shows how SA was retrofit into a Z80-based personal computer to allow testing and troubleshooting on the manufacturer's production line, customer service center, and eventually at their distributor's service centers. It shows the details of circuit stimulus features added for Signature Analysis testing of ROM, RAM and I/O in terms of hardware, software and test connections. It includes stimulus test loop flow charts and Z80 assembly code listings; START, STOP and CLOCK connections and waveforms; and a discussion of the tradeoffs associated with retrofitting SA into this product.

ABOUT OTHER PUBLICATIONS

Application Note 222-0, HP publication 02-5952-7593, is a complete index to the latest Signature Analysis Publications.

It lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests from how to design or retrofit Signature Analysis into digital systems to the cost reductions that can be expected in production test and field service by doing so.

It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

CONTENTS

SECTION A—INTRODUCTION

The Signature Analysis Technique

By designing or retrofitting the Signature Analysis (SA) technique into a microprocessor-based digital product, a manufacturer can provide simplified field service and production line procedures for component level repair of the product using a Signature Analyzer for troubleshooting. Use of a Signature Analyzer requires that some test features be designed or retrofitted into the product to be tested. This article assumes some familiarity with these features. Additional Hewlett-Packard Application Notes on Signature Analysis[1] provide the basics on how to add these features to a product to be serviced with a Signature Analyzer.

The Application

This application shows how SA was retrofit into a personal computer to test and troubleshoot it on the manufacturer's production line, customer service centers, and eventually at their distributor's service centers. Here is the strategy for testing the computer on the production line where SA is used to troubleshoot a completely unknown board. This means:

- No shorts or opens testing is done on blank boards.
- No incoming inspection testing of RAMs is performed, including 4K and 16K dynamic RAMs.
- Visual inspection is used to find most process faults of an assembled board such as solder splashes and bridges, and incorrectly installed ICs and components. No ATE is used.
- Boards are powered-up in an unknown state immediately after assembly and inspection.
- When a board does not operate correctly when power is applied, diagnostics separate from SA are used in an attempt to isolate failures down to small blocks of the circuit (e.g. it will indicate that the failure seems to be in ROM or RAM or the supporting address decodes and control circuits), but not to the component or process fault level in most cases.
- SA is used to find the components or process faults using SA stimulus routines on those boards that the diagnostic routine can indicate where to begin troubleshooting.
- SA is also used to troubleshoot boards that won't allow the diagnostic to run using a technique called FREERUN.

The distributors still plan on board exchange with the customer's failed unit, then bringing the board back to the shop for repairs with SA. The SA documentation is not yet available to the customer so that he could make his own after-warranty repairs easier with SA.

The Computer

This article assumes a working knowledge of microprocessor-based systems and raster-scan CRT text generators. Hardware and software manuals on the computer are available from the manufacturer[2]. They include a theory of operation and detailed schematics.

Figures 1 and 2 show the block diagram and memory map of the standard computer unit. The standard computing system incorporates the circuits of the block diagram within a single housing around the keyboard. Optional equipment is available which expands the capabilities of the computer but exists as separate items. They are a CRT display, a line printer, a floppy disc, and a S-100 bus expansion module. SA has been retrofit into the standard computer unit. It has not yet been implemented in any of the optional peripherals.

The Article

This article shows the details of implementing SA into the standard computer unit in terms of the hardware, software and test connections. The figures show how SA was retrofit into the computer while the accompanying text discusses the major decisions and tradeoffs associated with retrofitting SA into the product and the effects some of those decisions had on the ease of troubleshooting the computer. The SA stimulus routines for the computer are effective in finding faults with SA. However, the way in which they are implemented is not necessarily the only way nor the most efficient way it could be done.

---

[1]Application Note 222, A Designer's Guide to Signature Analysis; 222-1, Implementing Signature Analysis for Production Testing; 222-2, Application Articles on Signature Analysis.

[2]The personal computer is called the SORCERER II and is manufactured by Exidy Corporation of Sunnyvale, California.

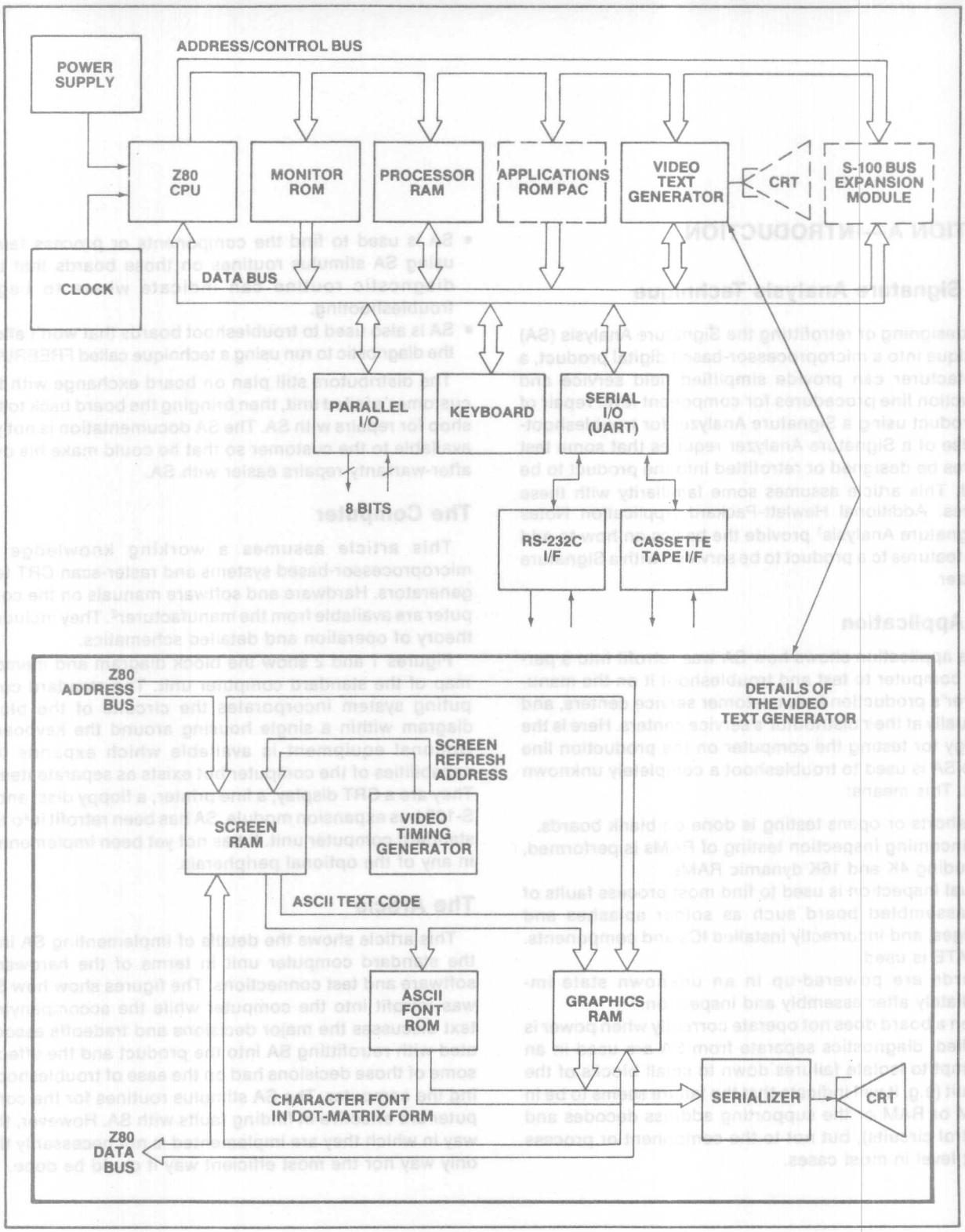


Figure 1. This personal computer contains a Z80 microprocessor, an operating system contained in MONITOR ROM, space for user created programs in PROCESSOR RAM, and a slot on the side of the computer to insert a ROM PAC that contains ROM-based applications programs. Also included is a VIDEO TEXT GENERATOR that outputs to a CRT for display. It includes memory space for ASCII text characters in SCREEN RAM, character font in ASCII FONT ROM, user defined graphics font in GRAPHICS RAM, and discrete VIDEO TIMING GENERATOR circuits. The interfaces to the computer consist of two SERIAL I/O channels for a cassette tape unit and a general purpose RS-232 link, both supported by a UART, an 8-bit PARALLEL I/O channel of discrete logic, a KEYBOARD with associated scan mechanism and a S-100 BUS EXPANSION port that is a buffered extension of the Z80 microprocessor bus. Optional equipment is outlined in dashed lines.

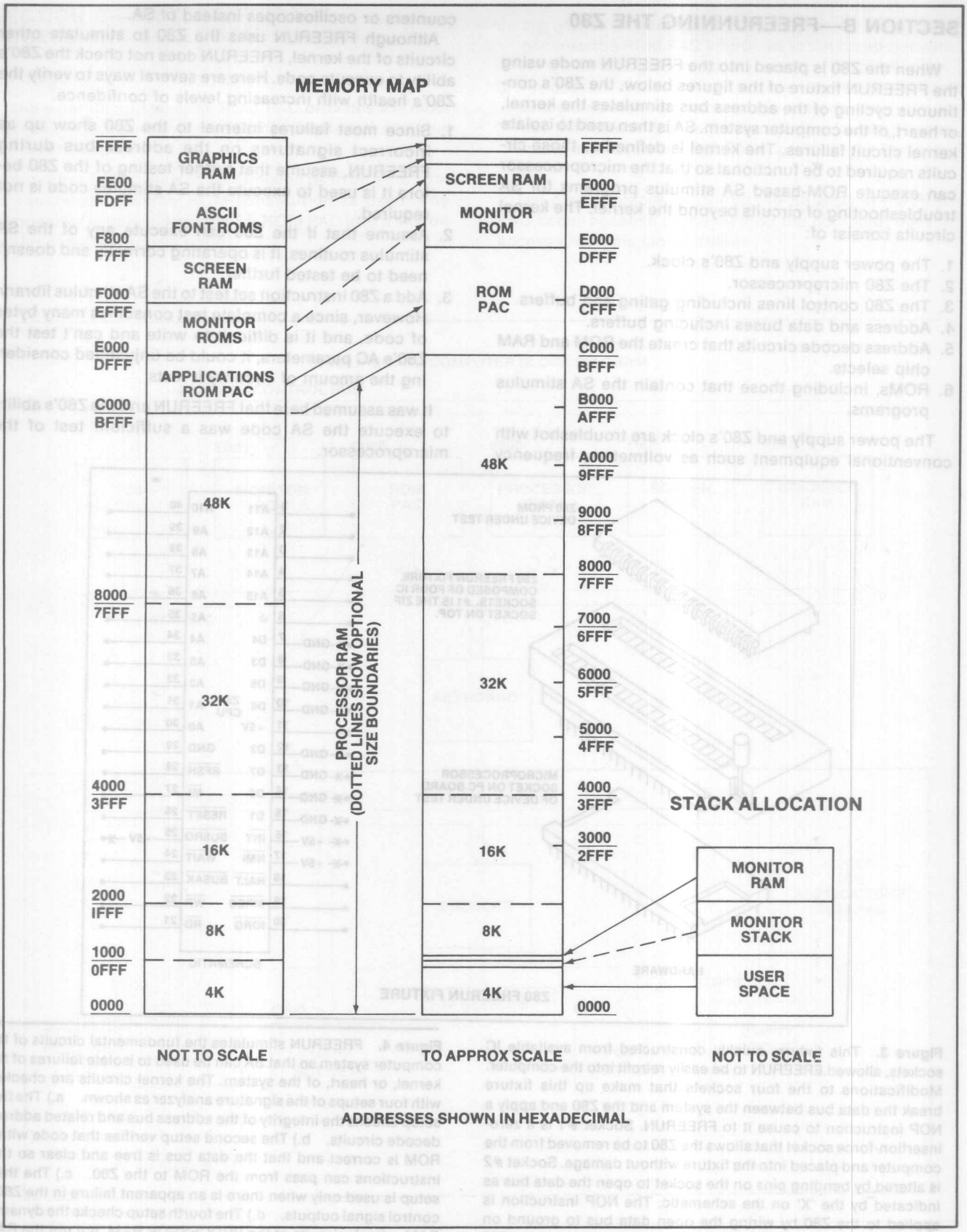


Figure 2. This diagram shows the Z80's 64K byte memory space divided among the circuits of the personal computer. All memory assignments are fixed except for the PROCESSOR RAM which can vary from 4K bytes to 48K bytes depending upon the user's memory requirements. The KEYBOARD, PARALLEL and SERIAL I/O devices reside within the input/output space of the Z80 while the S-100 BUS EXPANSION memory devices automatically map over the PROCESSOR RAM as needed.

SECTION B—FREERUNNING THE Z80

When the Z80 is placed into the FREERUN mode using the FREERUN fixture of the figures below, the Z80's continuous cycling of the address bus stimulates the kernel, or heart, of the computer system. SA is then used to isolate kernel circuit failures. The kernel is defined as those circuits required to be functional so that the microprocessor can execute ROM-based SA stimulus programs for SA troubleshooting of circuits beyond the kernel. The kernel circuits consist of:

1. The power supply and Z80's clock.
2. The Z80 microprocessor.
3. The Z80 control lines including gating and buffers.
4. Address and data buses including buffers.
5. Address decode circuits that create the ROM and RAM chip selects.
6. ROMs, including those that contain the SA stimulus programs.

The power supply and Z80's clock are troubleshot with conventional equipment such as voltmeters, frequency counters or oscilloscopes instead of SA.

Although FREERUN uses the Z80 to stimulate other circuits of the kernel, FREERUN does not check the Z80's ability to execute code. Here are several ways to verify the Z80's health with increasing levels of confidence.

1. Since most failures internal to the Z80 show up as incorrect signatures on the address bus during FREERUN, assume that further testing of the Z80 before it is used to execute the SA stimulus code is not required.
2. Assume that if the Z80 can execute any of the SA stimulus routines, it is operating correctly and doesn't need to be tested further.
3. Add a Z80 instruction set test to the SA stimulus library. However, since a complete test consumes many bytes of code, and it is difficult to write and can't test the Z80's AC parameters, it could be unjustified considering the amount of circuitry it tests.

It was assumed here that FREERUN and the Z80's ability to execute the SA code was a sufficient test of the microprocessor.

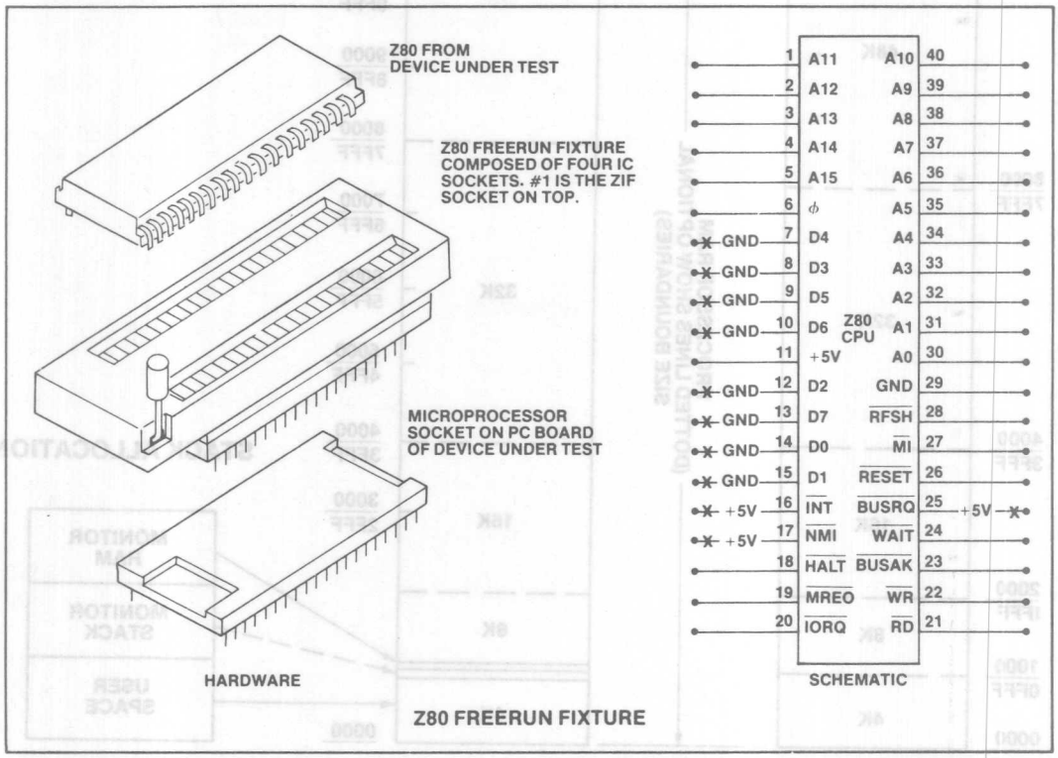
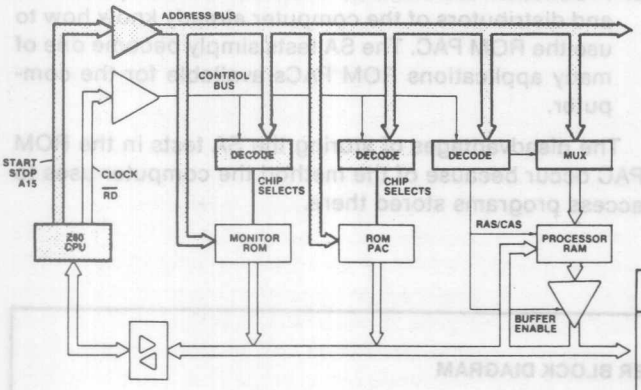


Figure 3. This fixture, quickly constructed from available IC sockets, allowed FREERUN to be easily retrofit into the computer. Modifications to the four sockets that make up this fixture break the data bus between the system and the Z80 and apply a NOP instruction to cause it to FREERUN. Socket #1 is a zero-insertion-force socket that allows the Z80 to be removed from the computer and placed into the fixture without damage. Socket #2 is altered by bending pins on the socket to open the data bus as indicated by the 'X' on the schematic. The NOP instruction is applied to the Z80 by wiring the open data bus to ground on socket #2. Similar treatment was done to several of the Z80 control pins by opening them and wiring to +5vdc or ground as required to force them to a known state during FREERUN independent of the computer's response.

Figure 4. FREERUN stimulates the fundamental circuits of the computer system so that SA can be used to isolate failures of the kernel, or heart, of the system. The kernel circuits are checked with four setups of the signature analyzer as shown. a.) The first setup checks the integrity of the address bus and related address decode circuits. b.) The second setup verifies that code within ROM is correct and that the data bus is free and clear so that instructions can pass from the ROM to the Z80. c.) The third setup is used only when there is an apparent failure in the Z80's control signal outputs. d.) The fourth setup checks the dynamic RAM refresh circuits of the PROCESSOR RAM, but not the RAM itself. The RAM cannot be troubleshot with FREERUN because it powers-up in a random state. There is no way to initialize it with the processor because the data bus has been opened up. All other circuits beyond the kernel are troubleshot using SA stimulus programs contained in ROM.

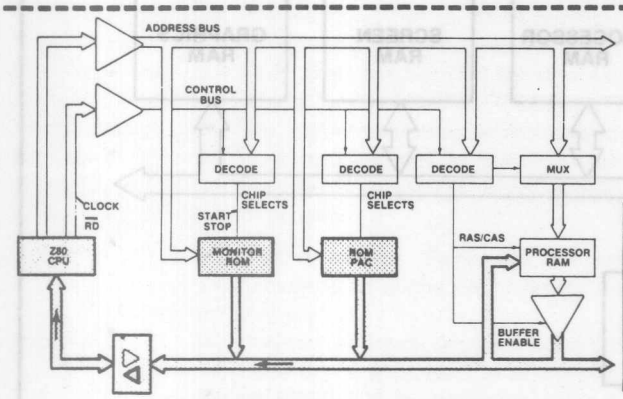


a. ADDRESS BUS AND DECODERS

SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	⌐	A15	5
STOP	⌐	A15	5
CLOCK	⌐	$\overline{RD}$	21

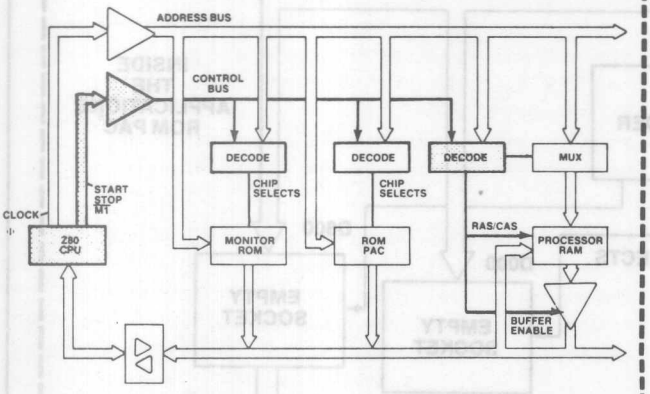
ADDRESS BUS SIGNATURE TABLE

ADDRESS	PIN	SIGNATURE	ADDRESS	PIN	SIGNATURE
A15	5	755U	A7	37	52F8
A14	4	3827	A6	36	UPFH
A13	3	3C96	A5	35	OAFA
A12	2	HAP7	A4	34	5H21
A11	1	1293	A3	33	7F7F
A10	40	HPP0	A2	32	CCCC
A9	39	2H70	A1	31	5555
A8	38	HC89	A0	30	UUUU



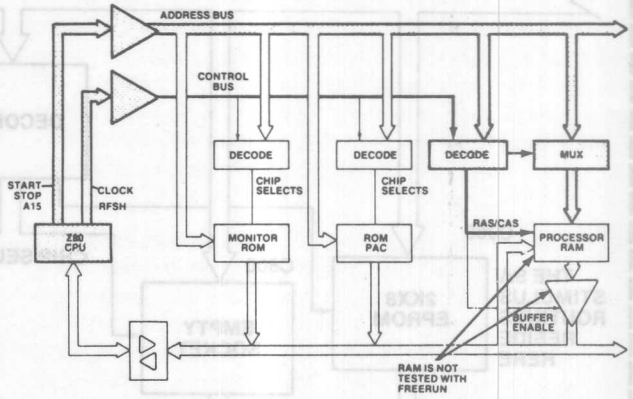
b. ROM CONTENTS AND DATA BUS

SIGNATURE ANALYZER		MONITOR ROM #1		MONITOR ROM #2	
INPUT	EDGE	SIGNAL	PIN	SIGNAL	PIN
START	⌐	CHIP SELECT	—	CHIP SELECT	—
STOP	⌐	CHIP SELECT	—	CHIP SELECT	—
CLOCK	⌐	Z80 $\overline{RD}$	21	Z80 $\overline{RD}$	21



SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	⌐	$\overline{MI}$	27
STOP	⌐	$\overline{MI}$	27
CLOCK	⌐	Φ	6

c. Z80 CONTROL LINES



SIGNATURE ANALYZER		Z80	
INPUT	EDGE	SIGNAL	PIN
START	⌐	A15	5
STOP	⌐	A15	5
CLOCK	⌐	$\overline{RFSH}$	28

d. DYNAMIC RAM REFRESH MULTIPLEXER

Figure 4

SECTION C—SA TEST STORAGE, ACCESS AND SELECTION

Storage

The external applications ROM PAC provides the most convenient and quick means to store the SA test stimulus for several reasons.

1. No other ROM needs to be removed as in most retrofit situations in which ROM-substitution is used. The operator simply inserts the SA ROM PAC into a slot on the computer to run them.
2. It's harder to damage a ROM PAC than it is a ROM because the ROM PAC interfaces to the computer with a PC board edge connector instead of fragile IC pins.
3. Production technicians, customer service personnel and distributors of the computer already know how to use the ROM PAC. The SA tests simply become one of many applications ROM PACs available for the computer.

The disadvantages of storing the SA tests in the ROM PAC occur because of the method the computer uses to access programs stored there.

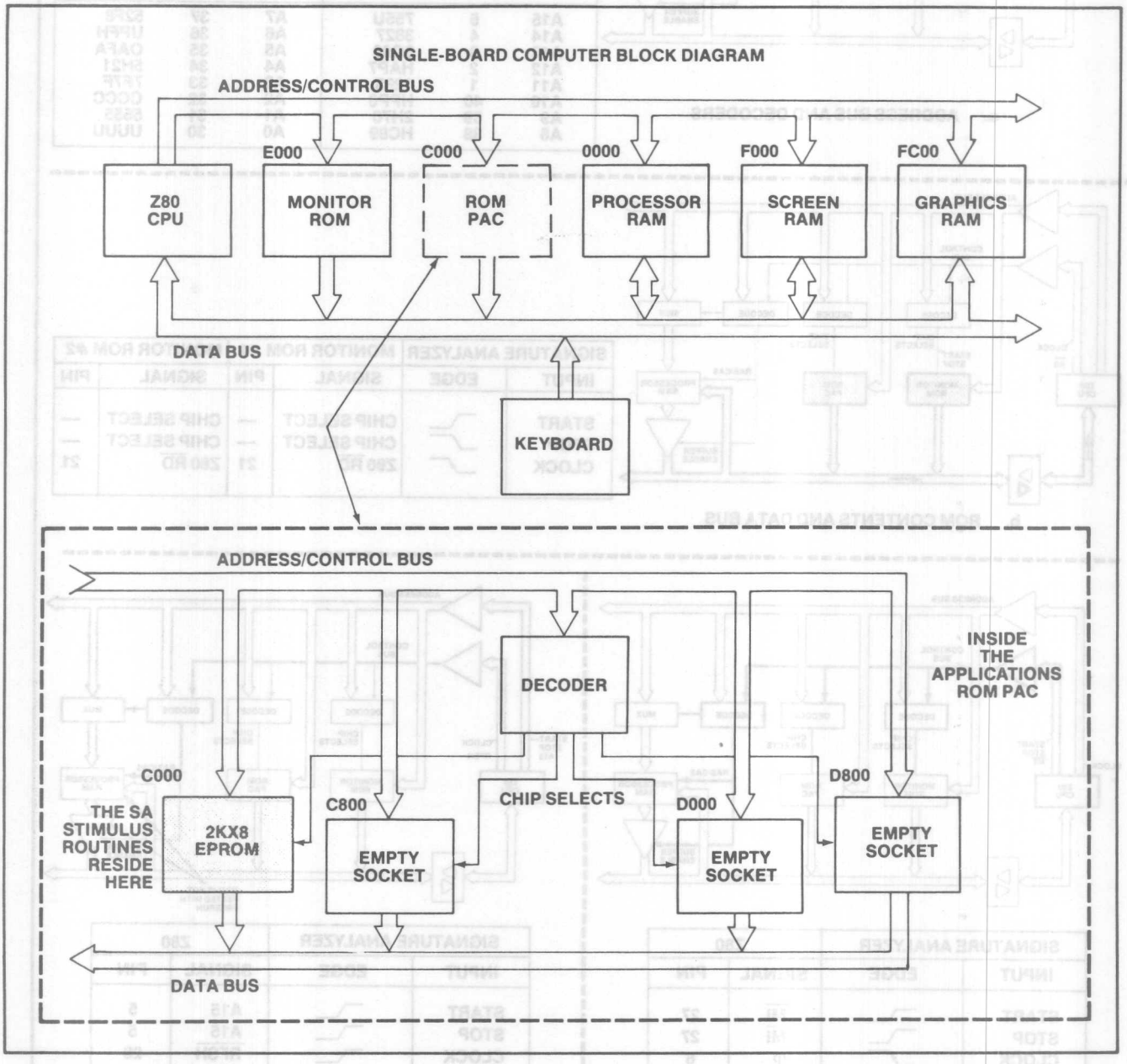


Figure 5. The SA stimulus routines (SA tests) are stored in a 2K × 8 EPROM within an applications ROM PAC that is inserted into the side of the computer. The SA tests are 944 bytes long of which 534 bytes are the ASCII characters of a test selection menu, 110 bytes are a branch table for test selection and 300 bytes form the actual SA tests. The program starts at the first address within the applications ROM PAC memory space, address C000H.

Access

When power is first applied to the computer, the Z80 CPU is reset to address 0000H and begins execution there. Since the ROM PAC resides at address C000H, a means is provided to cause the Z80 to jump there to begin execution of the SA tests. However, a ROM PAC is not always inserted into the computer. The program must also recognize the presence or absence of the ROM PAC and jump to the ROM PAC program if it's there by means of an operating system stored in the monitor ROMs.

The monitor ROMs reside at memory address E000H, not at address 0000H. (The first address executed by the Z80 at power-on.) To compensate, a special circuit that resets to zero at power-on temporarily maps the monitor ROMs to location 0000H. The first three locations of the monitor ROMs contain an unconditional jump instruction to location E003H. When the Z80 addresses E003H for the next instruction, the special circuit remaps the monitor ROMs back to their true address space of E000H to EFFFH so that further operating system code in the monitor ROMs can be executed.

The operating system then tries to establish a stack in any available functional RAM so that monitor subroutines can be accessed by the user. The operating system first checks processor RAM to see if enough locations behave like RAM. If the RAM is functional, the operating system establishes a stack and proceeds to do several housekeeping chores to set up the computer for interaction by the user. If processor RAM is not functioning, the computer will continue to search memory for the next available RAM locations (e.g. SCREEN or GRAPHICS RAM) until a stack can be established. If no RAM is functional, the operating system will continue to search forever and never release control to the user.

If a stack can be established, the operating system then checks for the presence of a ROM PAC to see if program execution should continue there. The process of determining this requires several subroutines within the operating system that uses the stack. Finally, the operating system sees that the SA ROM PAC has been inserted and jumps to the first location (address C000H) to put up the SA test selection menu. Once the SA tests have been selected and are executing, neither the operating system nor the stack are used.

With this background of how the SA tests are accessed, we can now discuss the tradeoffs of storing the SA tests in the ROM PAC versus storing them in a ROM which can be substituted in place of the monitor ROMs. (Substitution is the more common approach to retrofitting SA.) By placing the programs in the ROM PAC, all kernel circuits must be functional in order to run any SA tests. FREERUN is used to check them when they aren't. However, this application also contains circuits that cannot be troubleshot with FREERUN, but must also be functional. They are:

1. RAM, so that the operating system can establish a stack. This includes RAM chip selects and support circuits.
2. The special circuit that maps the monitor ROM to location 0000H at power-on.

An assumption was made that one of the three sections of RAM (either PROCESSOR, SCREEN or GRAPHICS

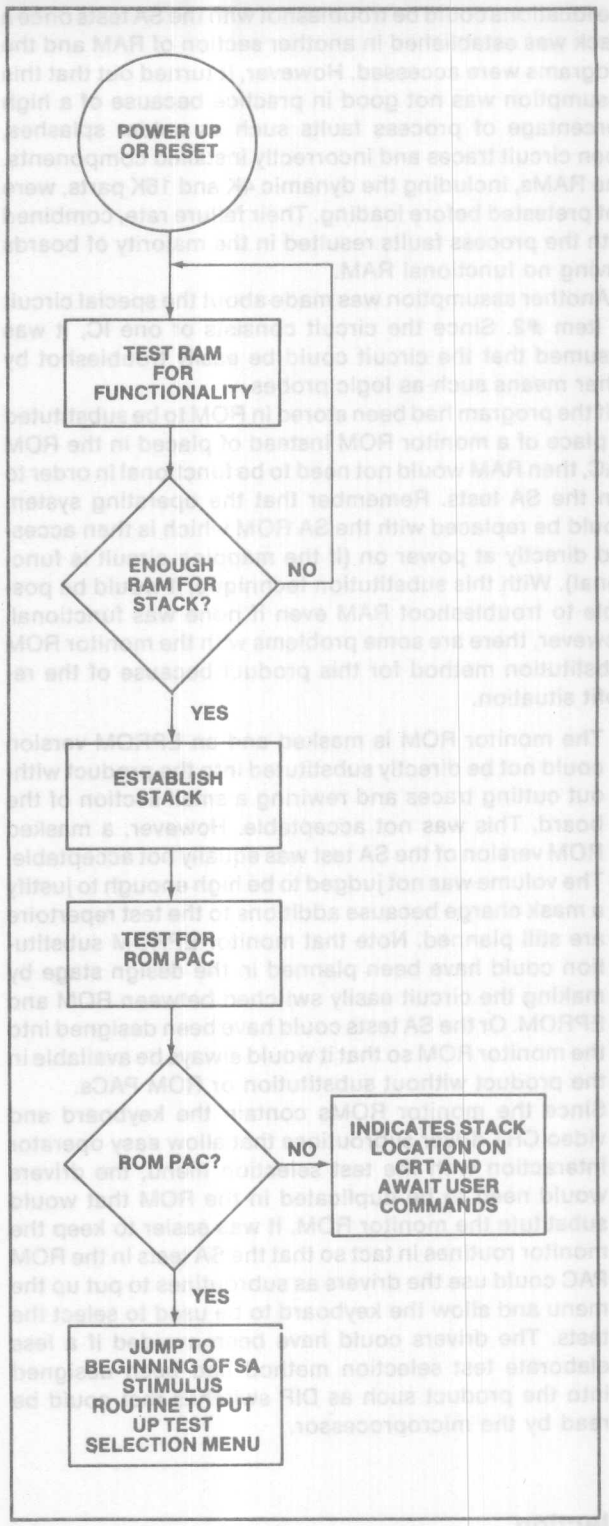


Figure 6. The SA tests are automatically accessed by the computer upon power-on when an SA applications ROM PAC is inserted. The system first goes through a power-up test of RAM to find a place for the operating system's stack to reside. The program then checks for the presence of a ROM PAC by checking the first few locations of ROM PAC memory space for non-zero entries. If the ROM PAC is present, the program then jumps to the first location of the ROM PAC to begin execution. The first steps of the SA tests put up a test selection menu on the CRT. See the next figure.

RAM) would be functional so that even bad RAM at one of the locations could be troubleshot with the SA tests once a stack was established in another section of RAM and the programs were accessed. However, it turned out that this assumption was not good in practice because of a high percentage of process faults such as solder splashes, open circuit traces and incorrectly installed components. The RAMs, including the dynamic 4K and 16K parts, were not pretested before loading. Their failure rate, combined with the process faults resulted in the majority of boards having no functional RAM.

Another assumption was made about the special circuit of item #2. Since the circuit consists of one IC, it was assumed that the circuit could be easily troubleshot by other means such as logic probes.

If the program had been stored in ROM to be substituted in place of a monitor ROM instead of placed in the ROM PAC, then RAM would not need to be functional in order to run the SA tests. Remember that the operating system would be replaced with the SA ROM which is then accessed directly at power on (if the mapping circuit is functional). With this substitution technique, it would be possible to troubleshoot RAM even if none was functional. However, there are some problems with the monitor ROM substitution method for this product because of the retrofit situation.

1. The monitor ROM is masked and an EPROM version could not be directly substituted into the product without cutting traces and rewiring a small section of the board. This was not acceptable. However, a masked ROM version of the SA test was equally not acceptable. The volume was not judged to be high enough to justify a mask charge because additions to the test repertoire are still planned. Note that monitor EPROM substitution could have been planned in the design stage by making the circuit easily switched between ROM and EPROM. Or the SA tests could have been designed into the monitor ROM so that it would always be available in the product without substitution or ROM PACs.
2. Since the monitor ROMs contain the keyboard and video CRT driver subroutines that allow easy operator interaction with the test selection menu, the drivers would need to be duplicated in the ROM that would substitute the monitor ROM. It was easier to keep the monitor routines in tact so that the SA tests in the ROM PAC could use the drivers as subroutines to put up the menu and allow the keyboard to be used to select the tests. The drivers could have been avoided if a less elaborate test selection method had been designed into the product such as DIP switches that could be read by the microprocessor.

Selection

The keyboard provides an easy means for the troubleshooter to quickly select a test. It is probably the easiest method when combined with the test selection menu on the CRT. However, should the keyboard fail, the operator cannot run any SA tests, including any that might help him troubleshoot the keyboard itself. Here it was decided that if the keyboard should fail, the operator would unplug the failed unit and exchange it with another one. However, this brings up two problems:

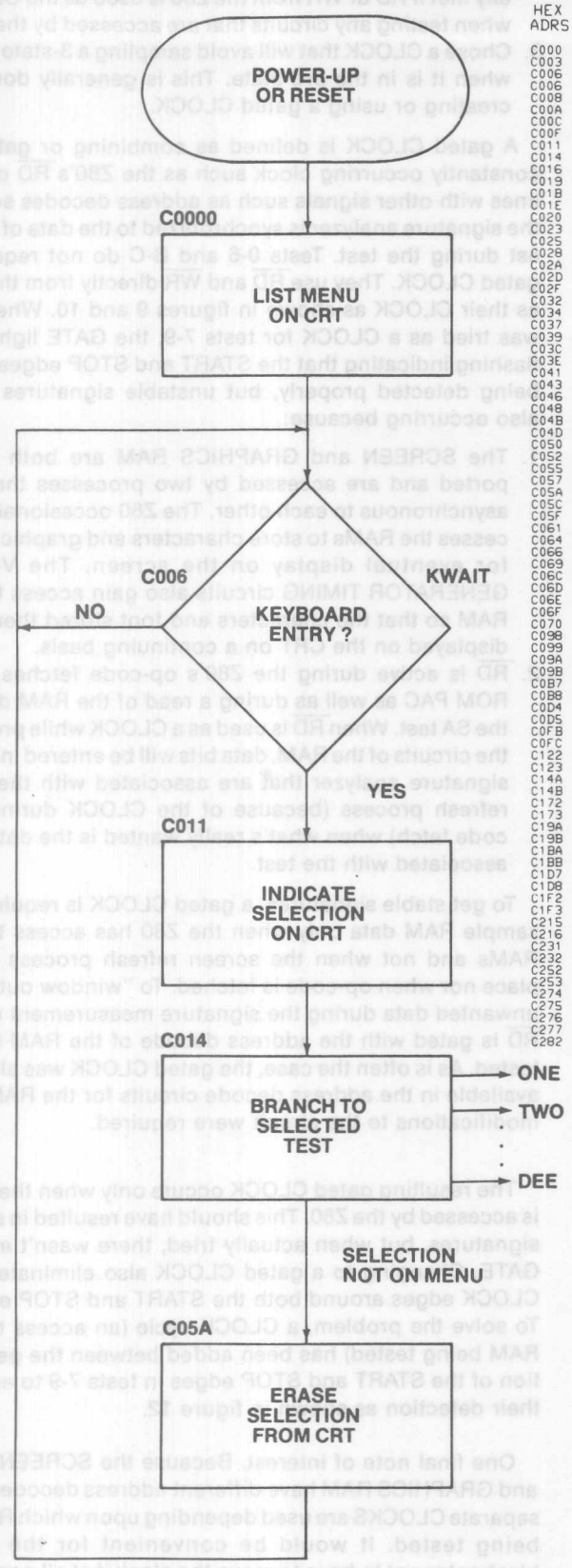
1. It assumes that all logic associated with the keyboard function exists on the keyboard PC board which is not the case here. The keyboard PC board contains only the key switches and one IC. Several other IC's associated with detecting a key closure are on the main board. The result is that exchanging the keyboard may not fix the problem.
2. Someone has to troubleshoot the keyboard when it fails, and SA cannot help if the tests are selected by means of the keyboard. Even FREERUN cannot help because the keyboard is treated as an I/O device that is not stimulated by FREERUN. It was assumed in this case that the keyboard could be fixed by other means.

```
SIGNATURE ANALYSIS TEST LOOPS

  0----MONITOR ROM #1
  1----MONITOR ROM #2
  2----4K PROCESSOR RAM, ROW #1
  3----4K PROCESSOR RAM, ROW #2
  4----16K PROCESSOR RAM, ROW #1
  5----16K PROCESSOR RAM, ROW #2
  6----16K PROCESSOR RAM, ROW #3
  7----BOTTOM SCREEN RAM
  8----TOP SCREEN RAM
  9----GRAPHICS RAM
  A----STATIC VIDEO PATTERN
  B----PARALLEL PORT
  C----SERIAL RS-232 PORT
  D----S-100 EXPANSION BUS
SELECT >>_
```

Figure 7. This menu appears on the CRT to prompt the user to select the number of the desired SA test with one keystroke. Once a test is selected it runs continuously and further keyboard entries are ignored. A reset from the keyboard, like power-on, stops the test and recalls the menu to the CRT for a new test selection. The details (code, flowcharts, and circuits) of tests 0-9 are shown in the remaining sections. Tests A-C are in separate sections with headlines that match the circuit they test and the menu number. For example, the PARALLEL PORT test is found in the section headlined " PARALLEL PORT , TEST B". TEST D is not implemented at this time.

Figure 8. Here is the flowchart and Z80 assembly language code that allows program selection. These are the mnemonics particular to the computer's assembler. PRTX is a monitor subroutine which transfers the ASCII characters of the menu from SA ROM to the CRT, starting at "MENU" (address C06CH) to "DEFB" (address C282H). Subroutine KEYBRD places a key entry into the accumulator with the zero flag reset. If no key is pressed, the zero flag is set. VIDEO writes the key entry on the CRT by the menu prompt "SELECT >>". If the entry matches a test number, the program branches to the test with the same label. (e.g. 3 branches to "THREE".) Invalid entries are erased from the CRT and the program returns to wait for a new key.




```
HEX
ADRS  CONTENTS  LABEL    INSTRUCTION

C000  216C00            LD    HL,MENU
C003  CDBAE1            CALL  PRTX
C006            KYWAIT:
C006  DBFE              IN    A,(0FEH)
C008  CB6F              BIT   5,A
C00A  28FA              JR    Z,KYWAIT-$
C00C  CD1BE0            CALL  KEYBRD
C00F  28F5              JR    Z,KYWAIT-$
C011  CD1BE0            CALL  VIDEO
C014  FE30              CP    30H
C016  CA83C2            JP    Z,ZERO
C019  FE31              CP    31H
C01B  CABCC2            JP    Z,ONE
C01E  FE32              CP    32H
C020  CAABC2            JP    Z,TWO
C023  FE33              CP    33H
C025  CAB4C2            JP    Z,THREE
C028  FE34              CP    34H
C02A  CABDC2            JP    Z,FOUR
C02D  FE35              CP    35H
C02F  CAC6C2            JP    Z,FIVE
C032  FE36              CP    36H
C034  CACFC2            JP    Z,SIX
C037  FE37              CP    37H
C039  CAD8C2            JP    Z,SEVEN
C03C  FE38              CP    38H
C03E  CAE1C2            JP    Z,EIGHT
C041  FE39              CP    39H
C043  CAEAC2            JP    Z,NINE
C046  FE41              CP    41H
C048  CA4AC3            JP    Z,AY
C04B  FE42              CP    42H
C04D  CA6DC3            JP    Z,BEE
C050  FE43              CP    43H
C052  CA7FC3            JP    Z,SEE
C055  FE44              CP    44H
C057  CAAFC3            JP    Z,DEE
C05A  3E08              LD    A,8
C05C  CD1BE0            CALL  VIDEO
C05F  3E20              LD    A,' '
C061  CD1BE0            CALL  VIDEO
C064  3E08              LD    A,8
C066  CD1BE0            CALL  VIDEO
C069  C306C0            JP    KYWAIT
C06C  0C        MENU    DEFB  12
C06D  0D                DEFB  13
C06E  0D                DEFB  13
C06F  0D                DEFB  13
C070  20202020          DEFM  '
C094  0D                DEFB  13
C095  0D                DEFB  13
C096  0D                DEFB  13
C09B  20202020          DEFM  '                  0----MONITOR ROM#1
C0B7  0D                DEFB  13
C0B8  20202020          DEFM  '                  1----MONITOR ROM#2
C0D4  0D                DEFB  13
C0D5  20                DEFM  '                  2----4K PROCESSOR RAM, ROW#1
C0FB  0D                DEFB  13
C0FC  20202020          DEFM  '                  3----4K PROCESSOR RAM, ROW#2
C122  0D                DEFB  13
C123  20                DEFM  '                  4----16K PROCESSOR RAM, ROW#1
C14A  0D                DEFB  13
C14B  20202020          DEFM  '                  5----16K PROCESSOR RAM, ROW#2
C172  0D                DEFB  13
C173  20202020          DEFM  '                  6----16K PROCESSOR RAM, ROW#3
C19A  0D                DEFB  13
C19B  20202020          DEFM  '                  7----BOTTOM SCREEN RAM
C1BA  0D                DEFB  13
C1BB  20202020          DEFM  '                  8----TOP SCREEN RAM
C1D7  0D                DEFB  13
C1D8  20202020          DEFM  '                  9----GRAPHICS RAM
C1F2  0D                DEFB  13
C1F3  20202020          DEFM  '                  A----STATIC VIDEO PATTERN
C215  0D                DEFB  13
C216  20202020          DEFM  '                  B----PARALLEL PORT
C231  0D                DEFB  13
C232  20202020          DEFM  '                  C----SERIAL RS-232 PORT
C252  0D                DEFB  13
C253  20202020          DEFM  '                  D----S-100 EXPANSION BUS
C274  0D                DEFB  13
C275  0D                DEFB  13
C276  0D                DEFB  13
C277  2053454C          DEFM  ' SELECT>>'
C282  00                DEFB  0
```

SECTION D—THE HARDWARE AND SOFTWARE BEHIND START, STOP AND CLOCK

Creating START and STOP

For tests 0-9 and B-C, the START and STOP edges were created by writing code that controlled available hardware. This combination of hardware and software is called program controlled gating. Figures 9-12 show the program controlled gating used in the computer. The START, STOP and CLOCK connections for FREERUN and test #A are shown in the corresponding sections of this article because they are NOT examples of program control gating and do not require any code to be written. The discussion here deals with some of the considerations in choosing an I/O register's output as the START and STOP connections for program control gating in the computer.

Here are some general guidelines for creating START and STOP. Create successive START and STOP connections that build on the kernel. That is, considering the START and STOP connections used in FREERUN as the first and most basic set, then the second set should be able to be troubleshot with the set used in FREERUN. The third set should be able to be troubleshot with the second and so on. Here are some suggestions for building a set of START and STOP connections for the Z80 from FREERUN to more complicated circuits.

1st set: FREERUN connections. Generally the most significant address bit of the Z80. Used to troubleshoot the next set.

2nd set: Chip selects and address decodes within the memory space of the processor that are stimulated during FREERUN to allow troubleshooting of them with START and STOP of the first set. Also controlled by software to create the required START and STOP edges for troubleshooting the next set.

3rd set: Bits in I/O registers that can be troubleshot with an SA stimulus routine that uses the START and STOP connections of the 2nd set. When the 3rd set is used as START and STOP, the software sets and resets the bit to create the required edges.

START and STOP for this computer fall into the 1st and 3rd sets. The 3rd set cannot be troubleshot with FREERUN, nor is there another test that can be run should START and STOP fail. This has caused the troubleshooter to resort to other methods such as shotgunning when the circuits creating START and STOP fail.

Detecting START, STOP and DATA with the CLOCK

Here are some guidelines for choosing a CLOCK.

1. A clock edge must occur both before and after the START and STOP edges to assure detection and correct GATE action. Be sure this is met when gated CLOCKs (defined below) are used.
2. The CLOCK must be synchronous to the START, STOP and DATA inputs it samples. This guideline is generally met if $\overline{RD}$ or $\overline{WR}$ from the Z80 is used as the CLOCK when testing any circuits that are accessed by the Z80.
3. Chose a CLOCK that will avoid sampling a 3-state node when it is in the 3rd state. This is generally done by creating or using a gated CLOCK.

A gated CLOCK is defined as combining or gating a constantly occurring clock such as the Z80's $\overline{RD}$ or $\overline{WR}$ lines with other signals such as address decodes so that the signature analyzer is synchronized to the data of interest during the test. Tests 0-6 and B-C do not require a gated CLOCK. They use $\overline{RD}$ and $\overline{WR}$ directly from the Z80 as their CLOCK as shown in figures 9 and 10. When $\overline{RD}$ was tried as a CLOCK for tests 7-9, the GATE light was flashing indicating that the START and STOP edges were being detected properly, but unstable signatures were also occurring because:

1. The SCREEN and GRAPHICS RAM are both dual-ported and are accessed by two processes that are asynchronous to each other. The Z80 occasionally accesses the RAMs to store characters and graphics font for eventual display on the screen. The VIDEO GENERATOR TIMING circuits also gain access to the RAM so that the characters and font stored there are displayed on the CRT on a continuing basis.
2. $\overline{RD}$ is active during the Z80's op-code fetches from ROM PAC as well as during a read of the RAM during the SA test. When $\overline{RD}$ is used as a CLOCK while probing the circuits of the RAM, data bits will be entered into the signature analyzer that are associated with the CRT refresh process (because of the CLOCK during op-code fetch) when what's really wanted is the data bits associated with the test.

To get stable signatures, a gated CLOCK is required to sample RAM data only when the Z80 has access to the RAMs and not when the screen refresh process takes place nor when op-code is fetched. To "window out" this unwanted data during the signature measurement cycle, $\overline{RD}$ is gated with the address decode of the RAM being tested. As is often the case, the gated CLOCK was already available in the address decode circuits for the RAM. No modifications to the circuit were required.

The resulting gated CLOCK occurs only when the RAM is accessed by the Z80. This should have resulted in stable signatures, but when actually tried, there wasn't even a GATE. Changing to a gated CLOCK also eliminated the CLOCK edges around both the START and STOP edges. To solve the problem, a CLOCK cycle (an access to the RAM being tested) has been added between the generation of the START and STOP edges in tests 7-9 to ensure their detection as shown in figure 12.

One final note of interest. Because the SCREEN RAM and GRAPHICS RAM have different address decodes, two separate CLOCKS are used depending upon which RAM is being tested. It would be convenient for the troubleshooter not to have to move the clock if at all possible. But because of the retrofit situation, it was not possible to do this here.

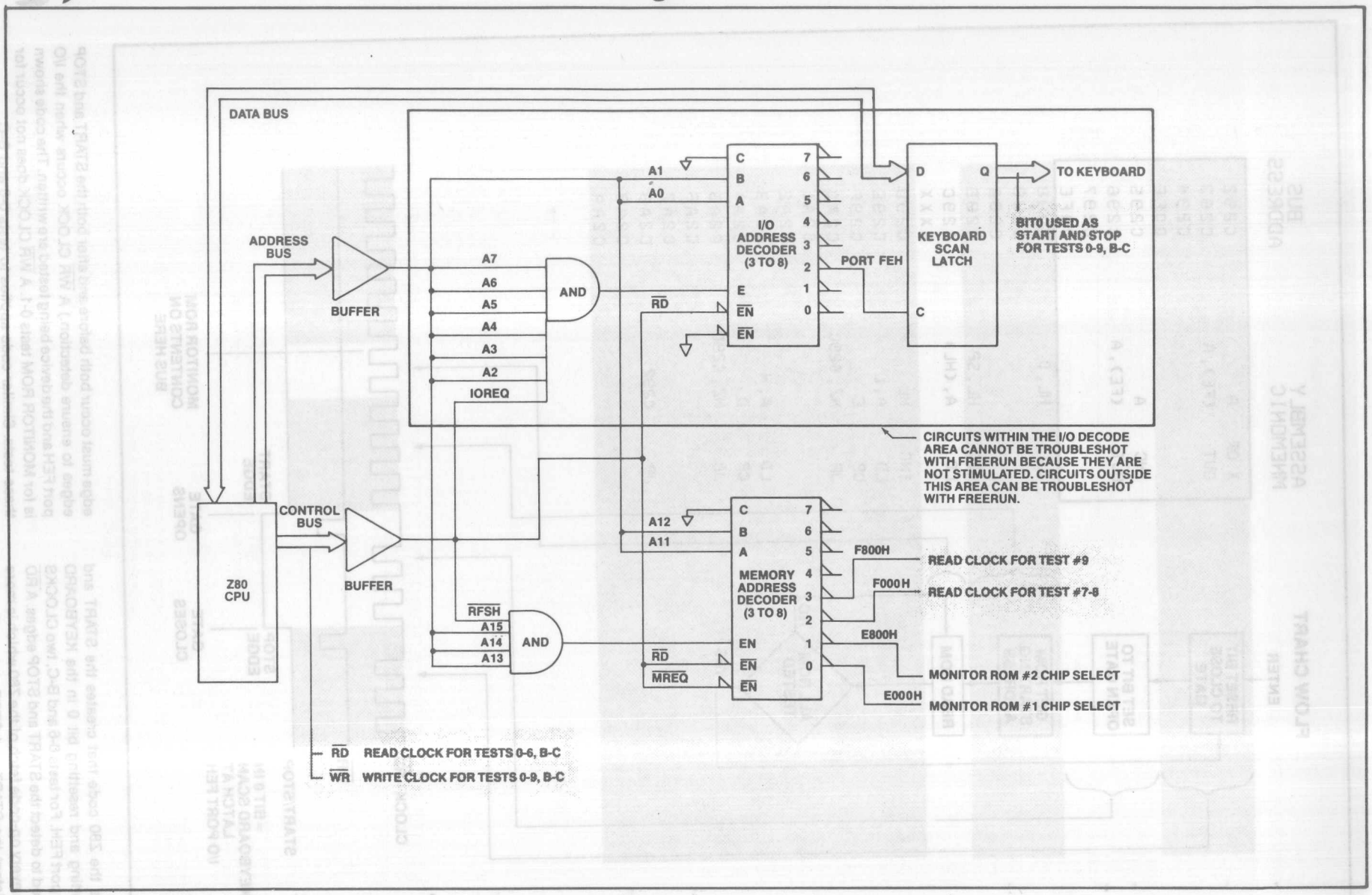


Figure 9. For all SA tests (except #A), START and STOP are both connected to bit 0 in the KEYBOARD SCAN LATCH addressed as I/O port FEH. The programs set and reset the bit to create edges that can be detected by the signature analyzer's START and STOP inputs. START is selected to trigger on a rising edge and STOP triggers on a falling edge. Setting the bit at the beginning of the test opens the GATE while resetting the bit immediately after the last test step closes the GATE. Figures 10 and 12 show the Z80 code that creates the START and STOP edges. The CLOCK connection depends on which test is run as discussed in figures 10-12. The START, STOP and CLOCK connections for test #A are shown in the separate description of that test.

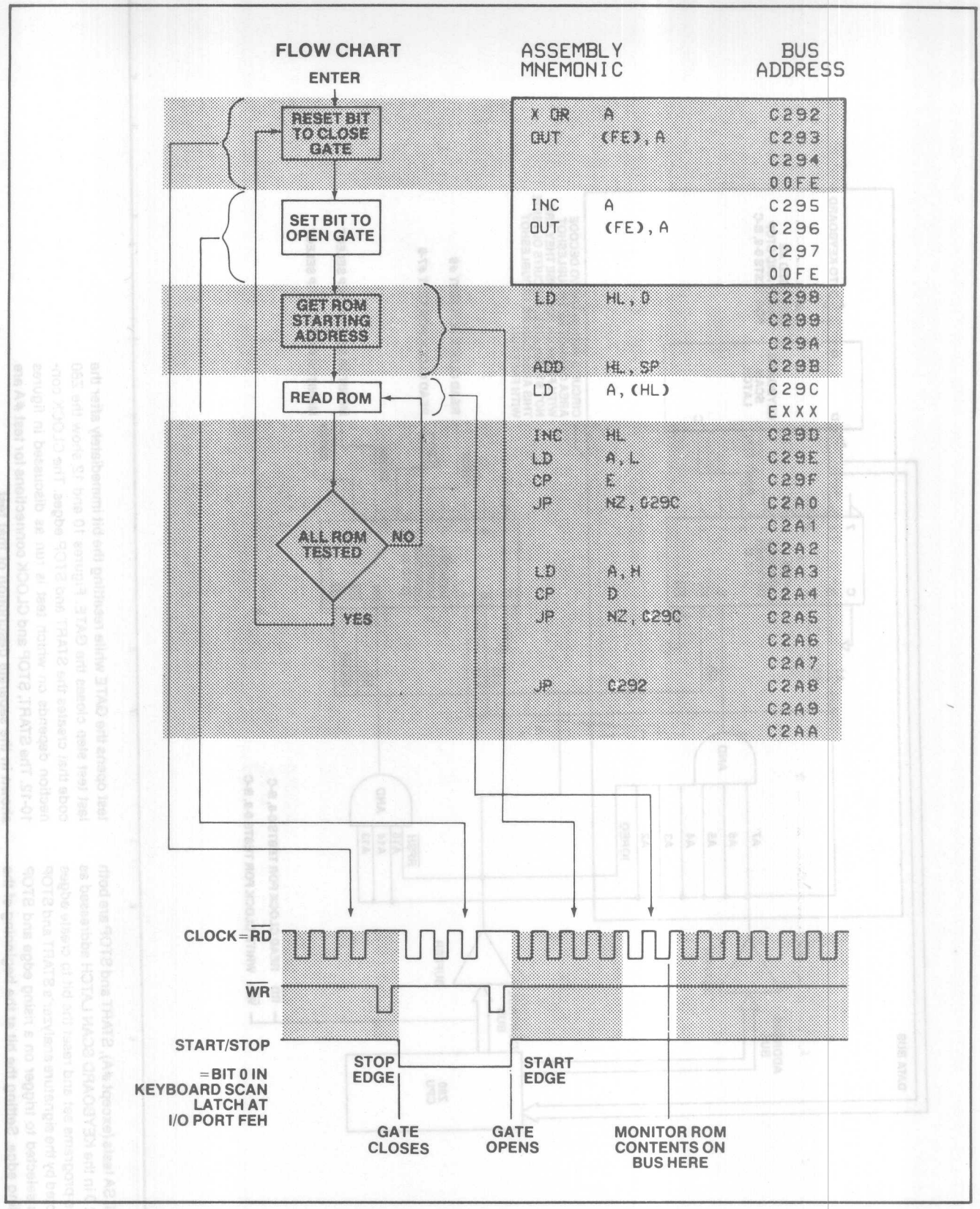


Figure 10. Here is the Z80 code that creates the START and STOP edges by setting and resetting bit 0 in the KEYBOARD SCAN LATCH at I/O port FEH. For tests 0-6 and B-C, two CLOCKS ($\overline{RD}$ and $\overline{WR}$) are used to detect the START and STOP edges. A $\overline{RD}$ CLOCK occurs with every op-code fetch of the Z80 which is more than adequate to detect the START and STOP edges. (A clock edge must occur both before and after both the START and STOP edges to ensure detection.) A $\overline{WR}$ CLOCK occurs when the I/O port FEH and the device being tested are written. The code shown is for MONITOR ROM tests 0-1. A $\overline{WR}$ CLOCK does not occur for these tests. Similar code applies to tests 0-6 and B-C.

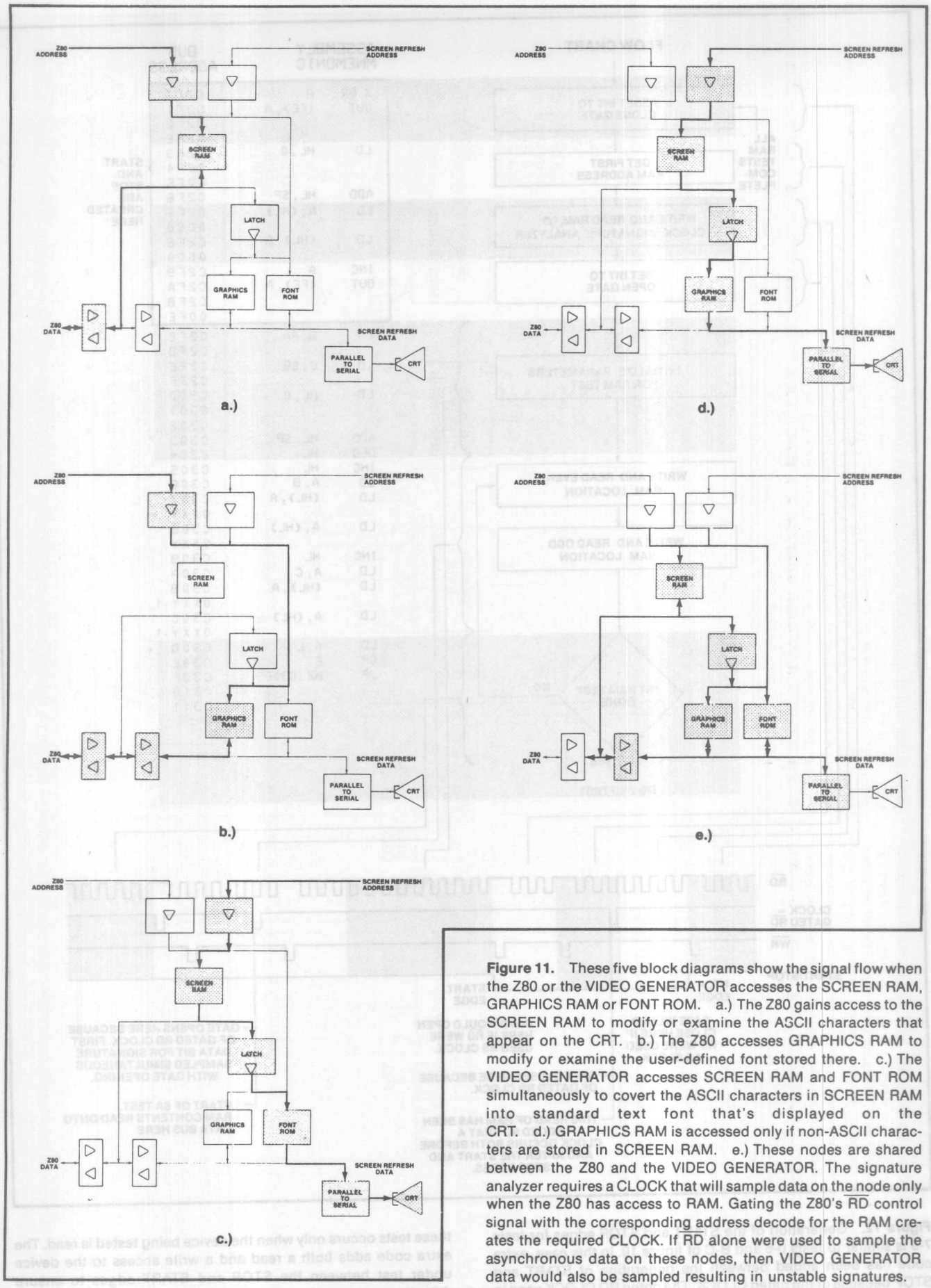


Figure 11. These five block diagrams show the signal flow when the Z80 or the VIDEO GENERATOR accesses the SCREEN RAM, GRAPHICS RAM or FONT ROM. a.) The Z80 gains access to the SCREEN RAM to modify or examine the ASCII characters that appear on the CRT. b.) The Z80 accesses GRAPHICS RAM to modify or examine the user-defined font stored there. c.) The VIDEO GENERATOR accesses SCREEN RAM and FONT ROM simultaneously to covert the ASCII characters in SCREEN RAM into standard text font that's displayed on the CRT. d.) GRAPHICS RAM is accessed only if non-ASCII characters are stored in SCREEN RAM. e.) These nodes are shared between the Z80 and the VIDEO GENERATOR. The signature analyzer requires a CLOCK that will sample data on the node only when the Z80 has access to RAM. Gating the Z80's $\overline{RD}$ control signal with the corresponding address decode for the RAM creates the required CLOCK. If $\overline{RD}$ alone were used to sample the asynchronous data on these nodes, then VIDEO GENERATOR data would also be sampled resulting in unstable signatures.

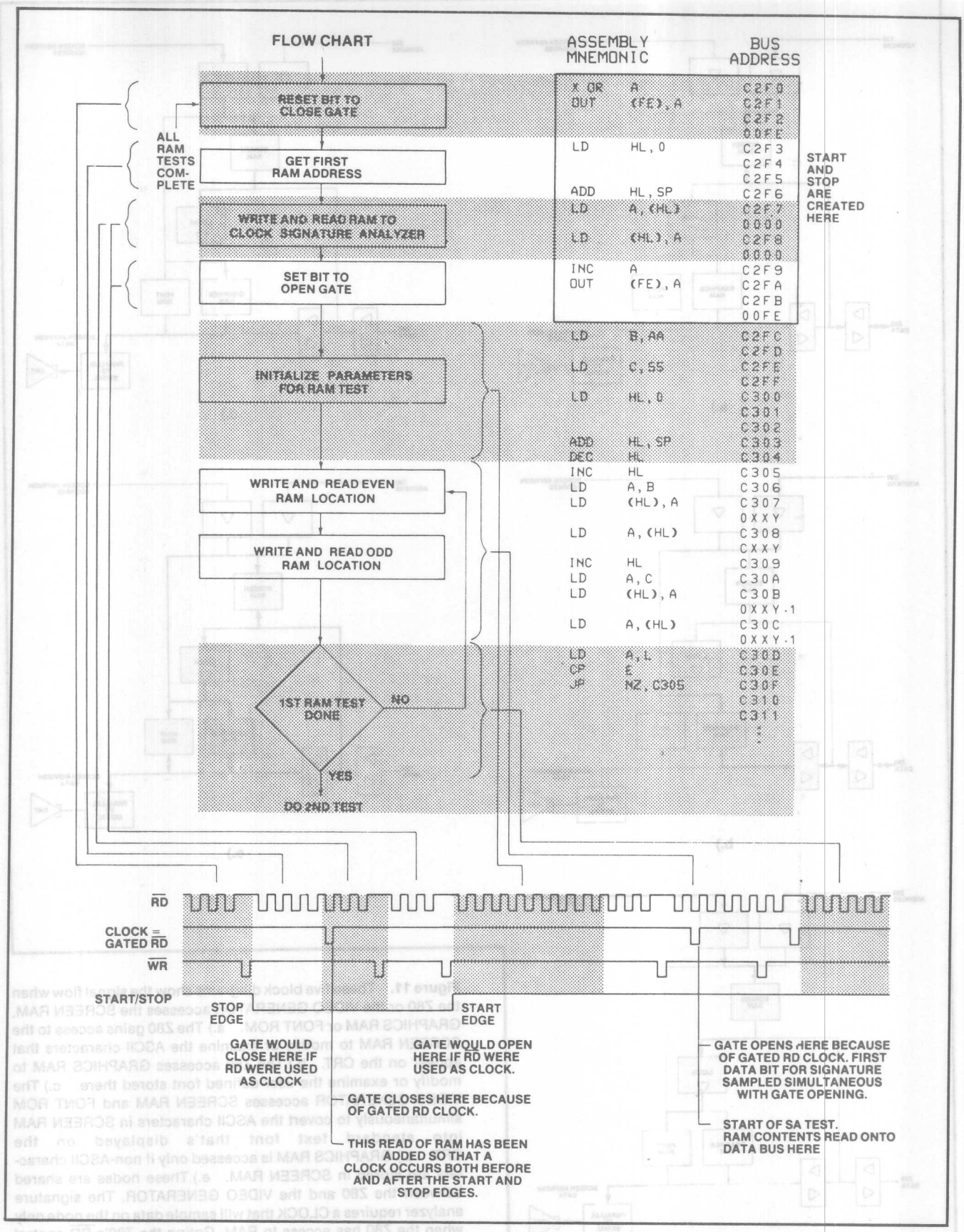


Figure 12. Generation of the START and STOP edges for tests 7-9 is similar to tests 0-6 and B-C of figure 10. In this case, extra code has been added between the generation of START and STOP because of the gated CLOCK. The gated $\overline{RD}$ CLOCK used in these tests occurs only when the device being tested is read. The extra code adds both a read and a write access to the device under test between the STOP and START edges to ensure detection.

SECTION E—FAULT ISOLATION OF BUSED DEVICES

SA Test Organization Makes the Difference

The way the SA tests are organized can make a difference in the ease of isolating a fault in a device that communicates over a data bus. SA tests are generally written two different ways depending upon the troubleshooting environment.

1. Go/No-go indication of all bused devices.

   All bused devices are tested at the same time within one SA test loop so that:
   a. If there is no fault, further testing is not required.
   b. If there is a fault, the failure is indicated to be within a limited area of the PC board.

2. Fault isolation of a specific bused device.

   The troubleshooter knows within which area of the board the fault lies, and now he's trying to locate the device or process fault causing the problem.

For example, consider the MONITOR ROM tests 0-1. They are written to test each MONITOR ROM separately. But imagine for a moment that both ROMs are tested within the same SA Loop. That is, the contents of both ROMs are read back onto the data bus between the same START and STOP. When the test is run, a go/no-go indication can be obtained from eight signatures taken on the data bus. Correct signatures indicate that everything is correct for both ROMs. Incorrect signatures would indicate a failure in one of the ROMs or in the supporting circuitry. But it's not known which ROM has failed until the contents of each ROM is individually examined with signatures. There are several ways to do this:

1. Remove all ROMs from their sockets or disable all chip selects. Then add one ROM at a time to the circuit until incorrect signatures reoccur on the data bus.
2. FREERUN the Z80 and "window" around each ROM's data by moving START and STOP to each chip select until incorrect signatures occur.
3. Create a separate test for each ROM so that only that ROM's data is placed on the bus. START, STOP and CLOCK can remain connected to the same signals if under program control.

Separate tests were written for several reasons:

1. The health of each ROM and supporting circuitry is determined quickly by running each test and taking only eight data bus signatures.
2. Diagnostics other than SA stimulus routines had already limited the failure to ROM. The troubleshooter was now looking for the device or process fault causing the failure.
3. It was simpler to select a new test to run than to remove parts from the board or move START, STOP and CLOCK around from chip select to chip select.
4. No modifications to the board or hardware could be added to allow the ROM chip selects to be disabled because of the retrofit situation.

Separate tests have also been written for the PROCESSOR RAM of tests 2-6. They are also implemented to find the one bused RAM out of several that could be causing the fault. Separating the tests also made easy testing of optional sizes of PROCESSOR RAM. Each socket for a RAM can accept a 4K or 16K dynamic RAM part, or no part at all, allowing PROCESSOR RAM to vary from 4K to 48K bytes total. Each variation of PROCESSOR RAM requires a new signature set. Since the PROCESSOR RAM tests are independent of the configuration, so is the signature documentation.

READ and WRITE as CLOCK Help Find the Faults

The SA test both reads and writes devices such as RAM, so that the Z80's $\overline{RD}$ and $\overline{WR}$ outputs can be used as the CLOCK to determine if the fault is caused by the RAM being incorrectly read or written. When $\overline{RD}$ is used as the clock, and signatures on the data bus are correct, the RAM is both being read and written correctly. When signatures are incorrect, the CLOCK is moved to $\overline{WR}$ to check the signatures on all inputs to the RAM including control lines. If all signatures are correct, the problem exists with reading RAM. The CLOCK is moved back to $\overline{RD}$ to isolate the problem caused by reading of the RAM. It may be the RAM itself, control line inputs to the RAM and the support circuits that generate them, or a process fault.

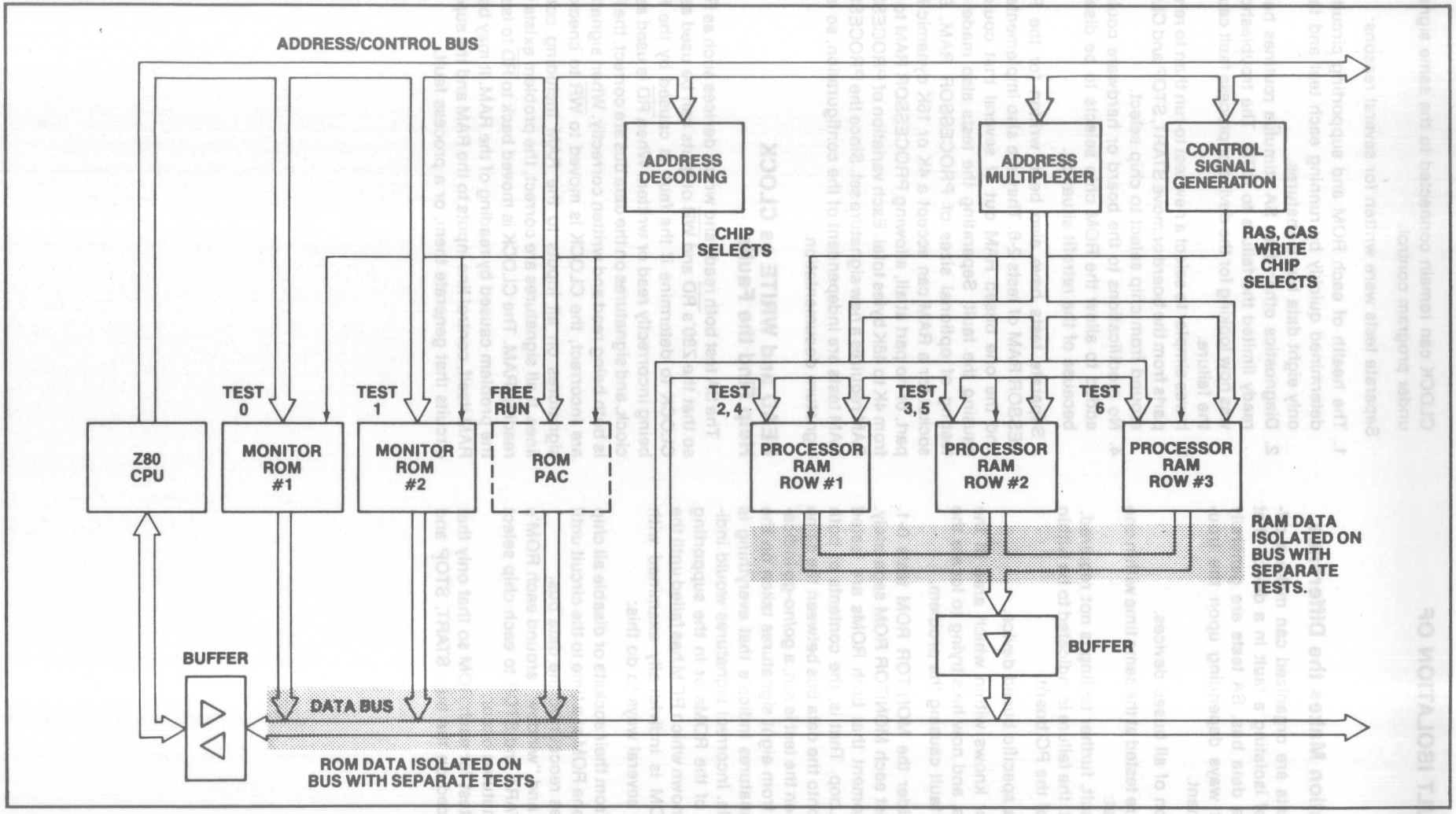


Figure 13. Each MONITOR ROM and each row of PROCESSOR RAM is tested separately to allow easier fault isolation of these bused devices. This ensures that the only data on any node comes from the device being tested and nothing else. (Except for the program that is executing from ROM PAC, which has to be good for the program to run. It is tested with FREERUN when it's bad.) A faulty device being tested is isolated by following incorrect signatures back to the source of the fault. Similarly, when data comes from devices that shouldn't be on the bus, it is detected as an incorrect signature and is also traced to the fault. $\overline{RD}$ and $\overline{WR}$ as CLOCKS also sort out faults. When incorrect signatures appear on the outputs of the RAM during a test using $\overline{RD}$ as a CLOCK, sometimes one or more RAM cell is bad, or control signals into the RAM are faulty (such as RAS, CAS, and CHIP SELECT), or the data was incorrectly written into RAM. To isolate, $\overline{WR}$ is used as a CLOCK to check signatures on the inputs that could affect writing of RAM (e.g. the data bus inputs and control lines such as RAS, CAS, WRITE and CHIP SELECT). If they have the correct signatures, the problem has to do with reading RAM.

SECTION F—STATIC VIDEO PATTERN, TEST A

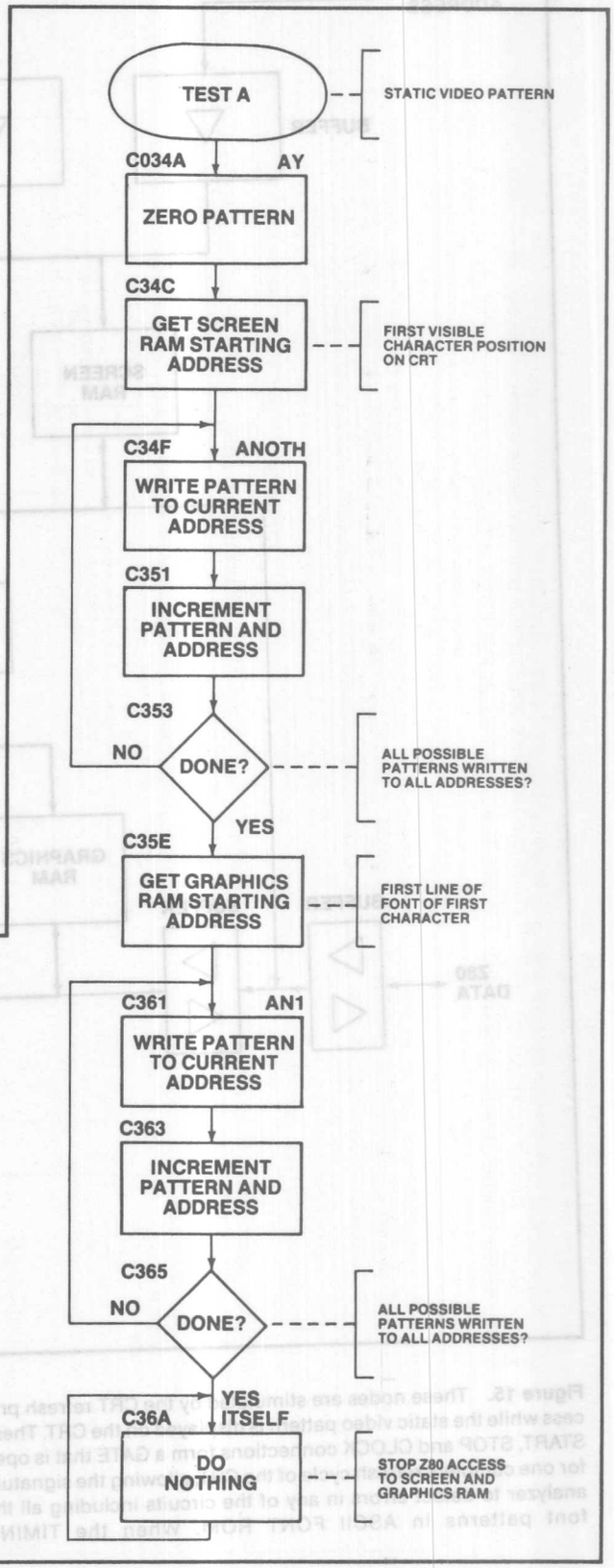


Figure 14. The static video pattern is written into RAM by this program. All possible characters and all possible patterns of user-defined font are written into SCREEN RAM and GRAPHICS RAM respectively. The program then enters a small loop that keeps the Z80 from further interaction with either RAM. This keeps the node activity limited to the CRT refresh process so that signatures are stable.

```
HEX
ADRS   CONTENTS   LABEL    INSTRUCTION

C34A              AY
C34A   0600                LD     B, 0
C34C   2180F0              LD     HL, 0F080H
C34F              ANOTH:
C34F   78                  LD     A, B
C350   77                  LD     (HL), A
C351   23                  INC    HL
C352   04                  INC    B
C353   7D                  LD     A, L
C354   B7                  OR     A
C355   C24FC3              JP     NZ, ANOTH
C358   7C                  LD     A, H
C359   FEF8                CP     0F8H
C35B   C24FC3              JP     NZ, ANOTH
C35E   21000FC             LD     HL, 0FC00H
C361              AN1:
C361   78                  LD     A, B
C362   77                  LD     (HL), A
C363   23                  INC    HL
C364   04                  INC    B
C365   7D                  LD     A, L
C366   B4                  OR     H
C367   C261C3              JP     NZ, AN1
C36A   C36AC3     ITSELF   JP     ITSELF
```

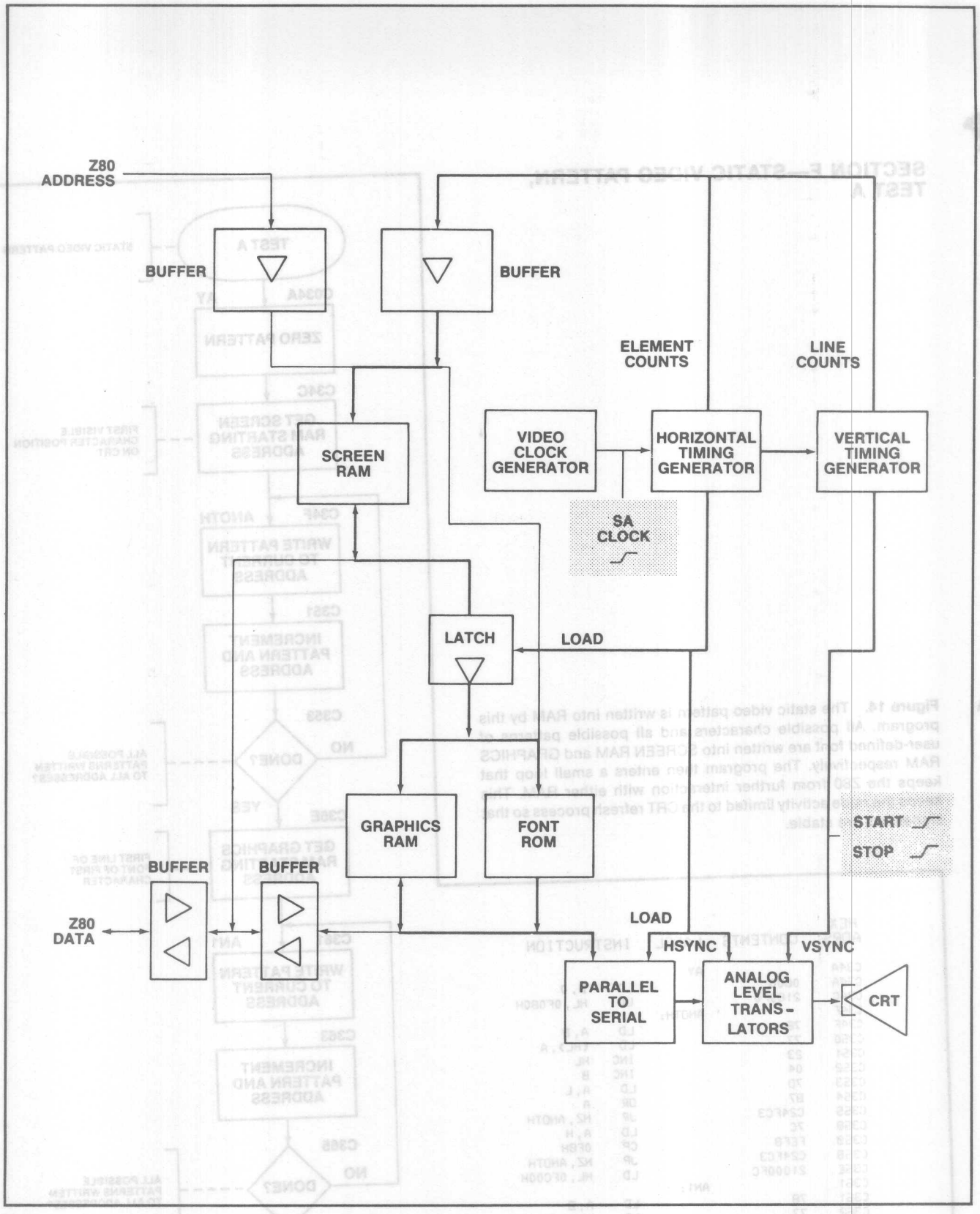


Figure 15. These nodes are stimulated by the CRT refresh process while the static video pattern is displayed on the CRT. These START, STOP and CLOCK connections form a GATE that is open for one complete refresh cycle of the CRT, allowing the signature analyzer to detect errors in any of the circuits including all the font patterns in ASCII FONT ROM. When the TIMING GENERATORS fail, START and STOP are no longer generated. In that case, START and STOP are moved to the connections shown in the next figures, closer to the kernel circuits of the VIDEO TEXT GENERATOR.

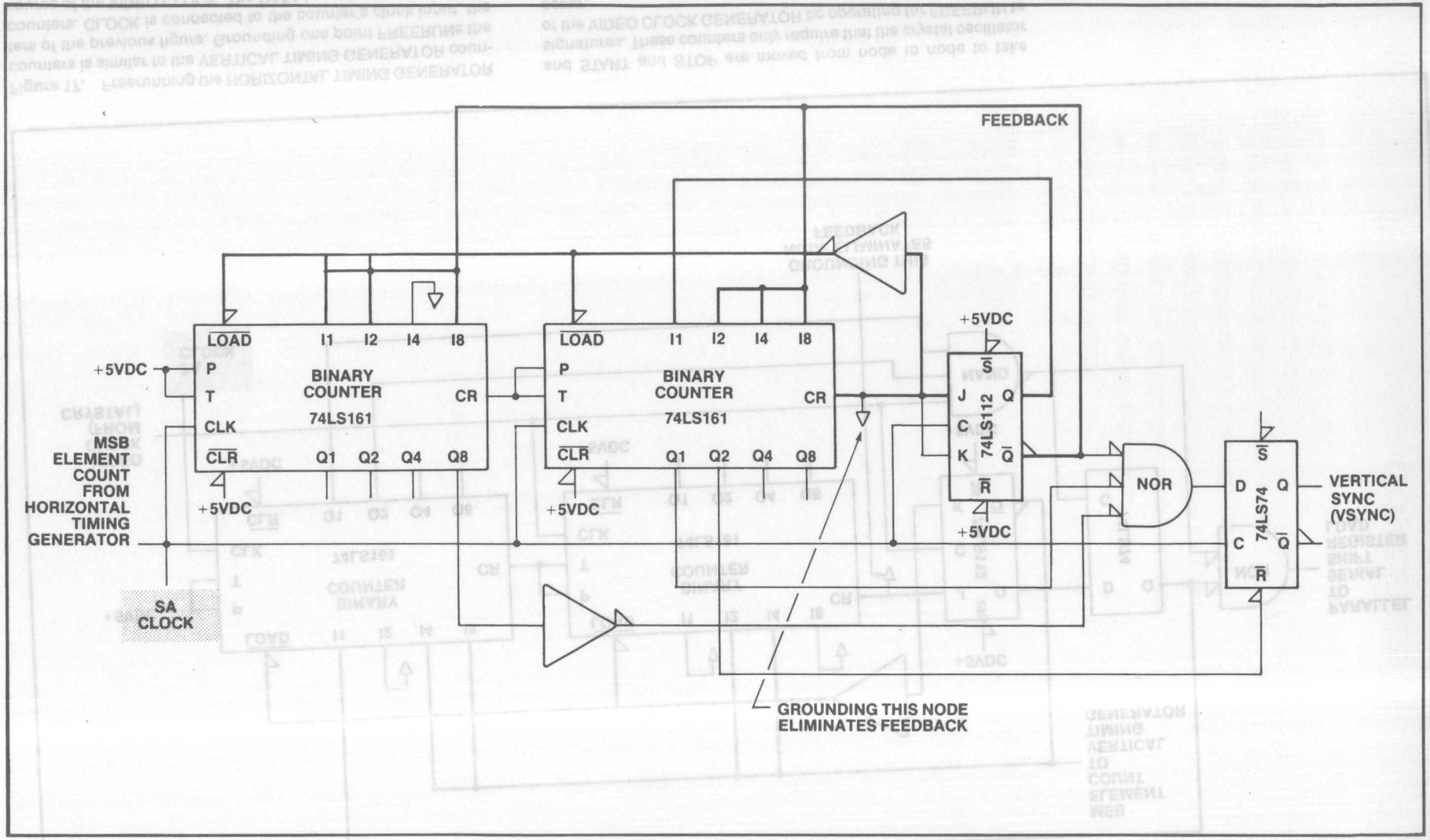


Figure 16. Grounding one point causes these VERTICAL TIMING GENERATOR counters to "FREERUN" through all possible states instead of counting CRT vertical lines. It does this by eliminating feedback loops. While grounding the output of a TTL device is not a recommended procedure, it was not possible to design-in a jumper that would disconnect the output from the circuit and ground the remaining inputs. To troubleshoot the freerunning counters, CLOCK is connected to the counter's clock input (the MSB of the HORIZONTAL TIMING GENERATOR of the next figure). The DATA input is placed on a source of logic high such as +5vdc. START and STOP are both moved to the circuit node under test to take a signature. The signature for any node then represents the number of CLOCK edges between START and STOP. If incorrect signatures occur for all nodes of these counters, then START, STOP and CLOCK are moved as shown in the next figure.

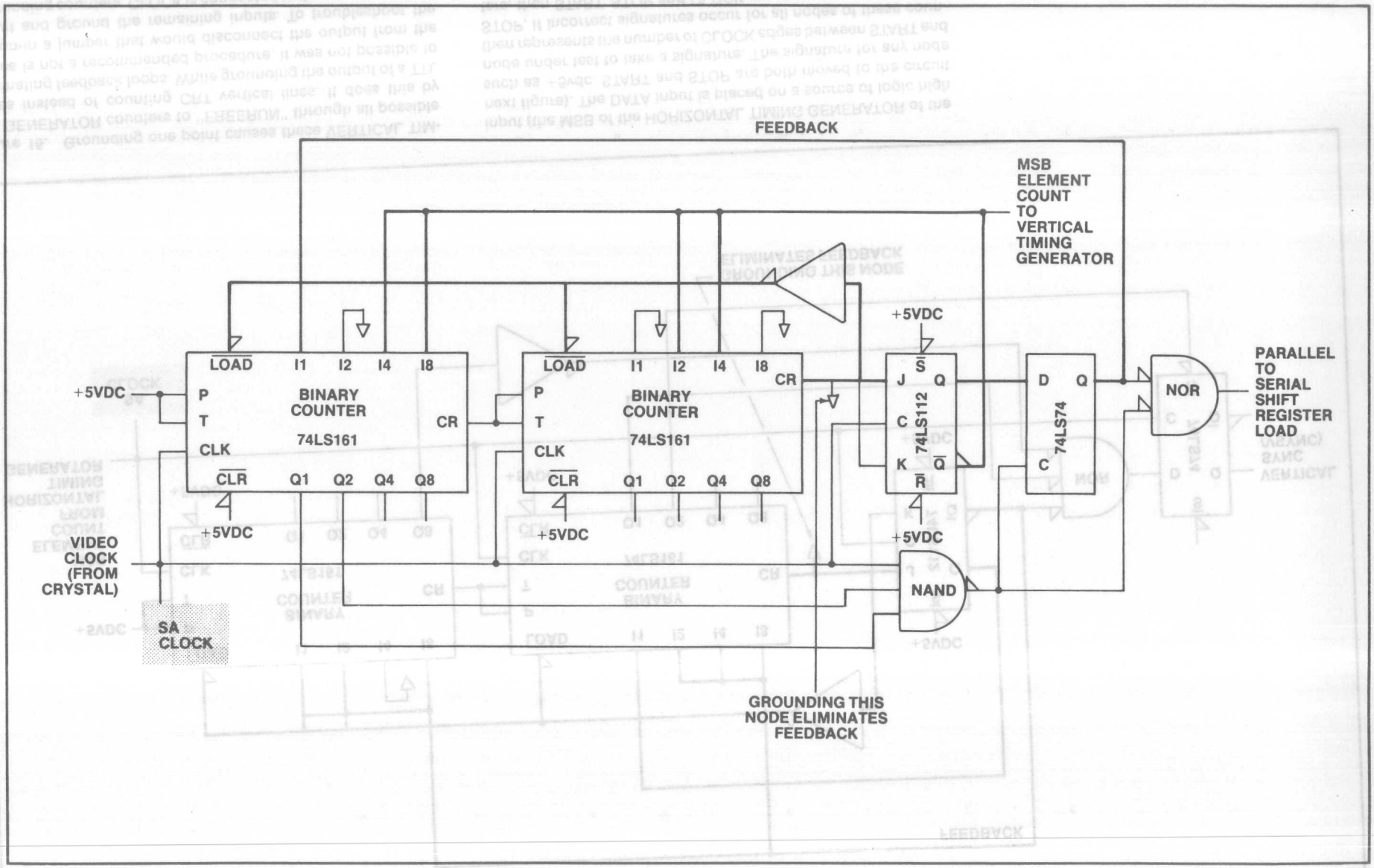


Figure 17. Freerunning the HORIZONTAL TIMING GENERATOR counters is similar to the VERTICAL TIMING GENERATOR counters of the previous figure. Grounding one point FREERUNs the counters. CLOCK is connected to the counter's clock input, the source of the VIDEO CLOCK. The DATA input is placed on +5vdc and START and STOP are moved from node to node to take signatures. These counters only require that the crystal oscillator of the VIDEO CLOCK GENERATOR be operating for FREERUN to occur.

SECTION G—PARALLEL PORT, TEST B

External hardware was required to stimulate the INPUT PORT. Wires on an external connector loop the patterns that are written to the OUTPUT PORT back to the INPUT PORT. Timing requirements of the HANDSHAKE control lines made it impossible to simply loop their outputs back to the inputs with similar wires on the connectors. It was decided not to add the ICs that would be required to fully stimulate the HANDSHAKE circuits. Two things determined this. First, the extra hardware would not be available to the distributors or field service personnel. Second, the HANDSHAKE circuits consist of one IC that can be easily troubleshot with other means such as logic probes.

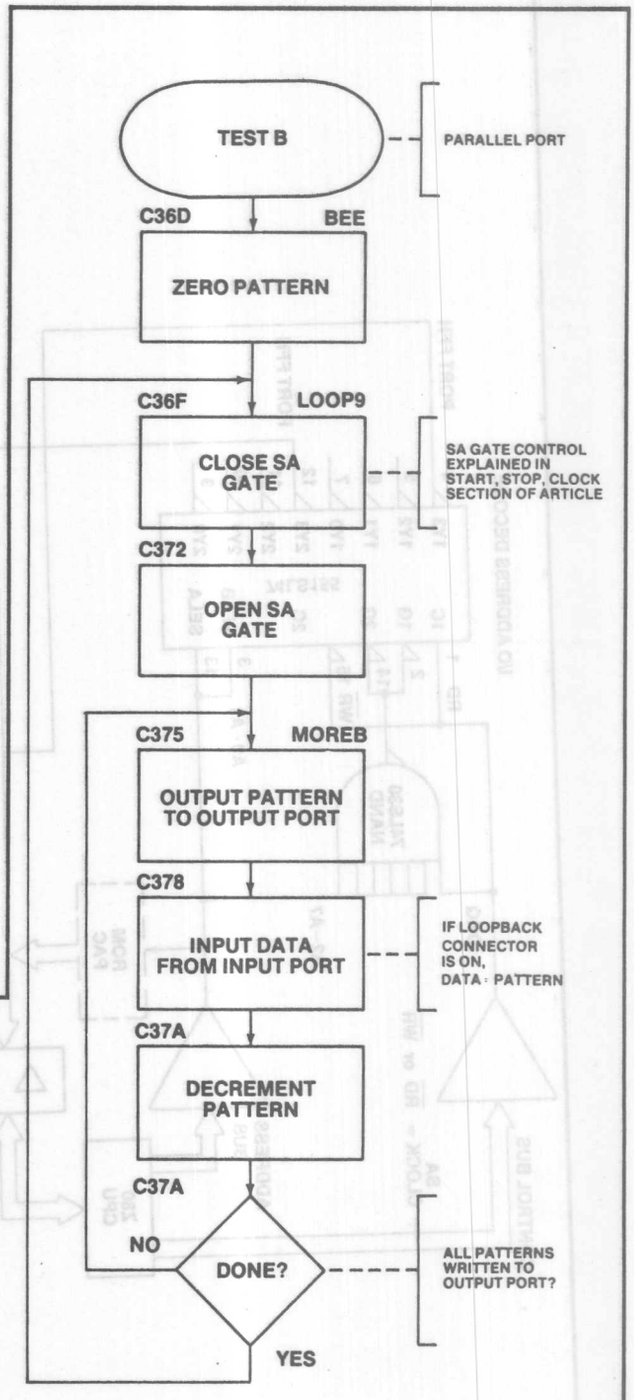


Figure 18. This program stimulates both the OUTPUT and INPUT PARALLEL PORTS by continuously writing all possible patterns to the OUTPUT PORT. The program also reads the INPUT PORT whether it is stimulated or not. The INPUT PORT is stimulated by looping the OUTPUT PORT back to the INPUT PORT using the connector shown in the following figure.

```
HEX
ADRS   CONTENTS   LABEL    INSTRUCTION

C36D              BEE:
C36D   0600                LD    B,0
C36F              LOOP9:
C36F   AF                  XOR   A
C370   D3FE                OUT   (0FEH),A
C372   3C                  INC   A
C373   D3FE                OUT   (0FEH),A
C375              MOREB:
C375   78                  LD    A,B
C376   D3FF                OUT   (0FFH),A
C378   DBFF                IN    A,(0FFH)
C37A   10F9                DJNZ  MOREB-$
C37C   D36FC3              JP    LOOP9
```

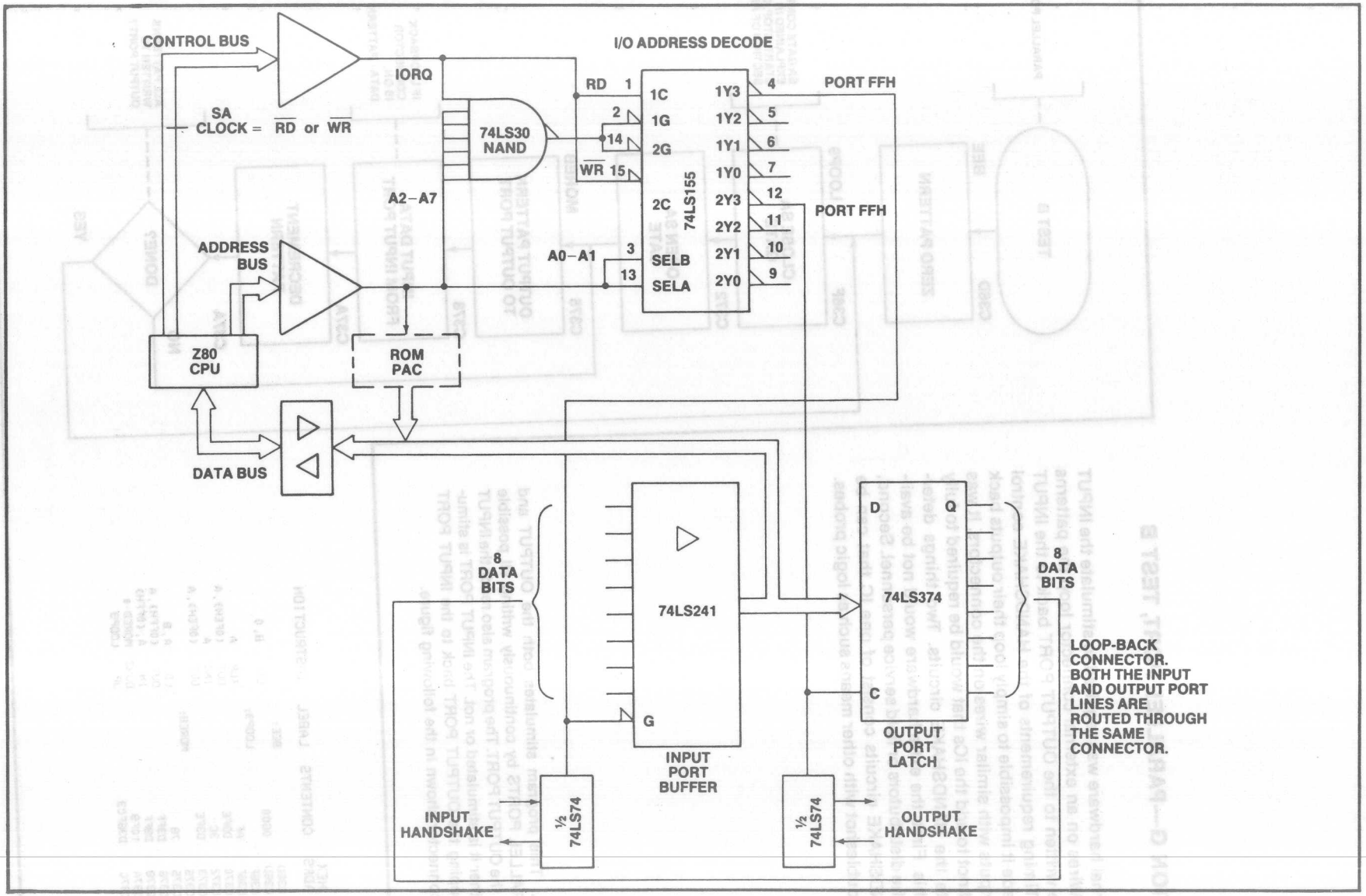


Figure 19. These circuits are stimulated by the PARALLEL PORT test. START and STOP are connected to a bit in the KEYBOARD SCAN LATCH and are controlled by the program as shown in the section called THE HARDWARE AND SOFTWARE BEHIND START, STOP AND CLOCK. With the loop-back connector on, signatures are taken on the data bus using $\overline{RD}$ as a CLOCK. If signatures are correct, then both the OUTPUT and INPUT PORT are operating correctly. If signatures are incorrect, the connector is removed and signatures are taken on the OUTPUT PORT using $\overline{WR}$ as the CLOCK. Repairs are made as required. The loopback connector is then replaced to check the INPUT PORT for problems associated with reading it. The text explains why the HANDSHAKE circuitry is not stimulated by this test.

SECTION H—SERIAL RS-232 PORT, TEST C

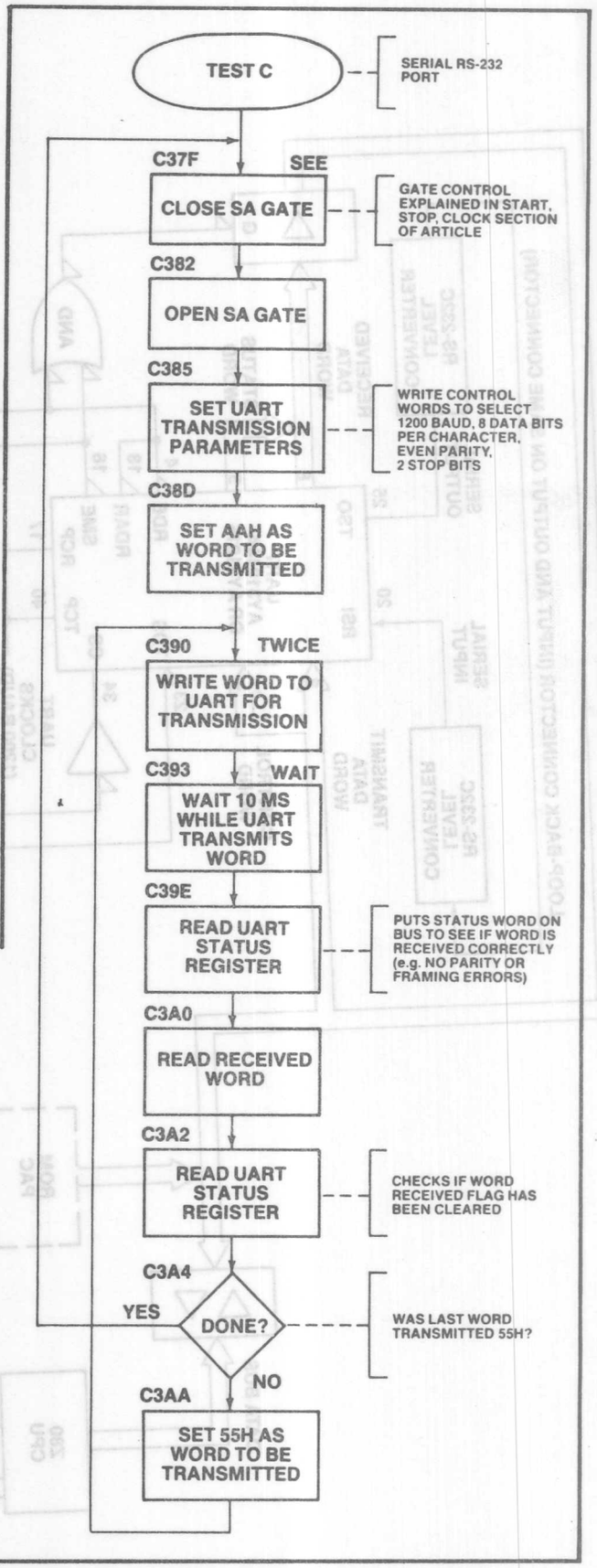
This test stimulates the UART for SA troubleshooting. UART's are generally considered to be test problems because of the lack of synchronization between the parallel side and the serial side of the part. However, the UART in this circuit (and in most applications) can be checked with SA as follows.

1. The serial output is connected to the serial input to take advantage of the loop-back technique for port testing, described in Section G. Figure 21 shows the UART loop-back configuration.

2. The stimulus program for Test C writes checkerboard patterns to the UART, then reads them back onto the data bus, allowing a fixed time for loop-back transmission of the serial words. This program is described in Figure 20.

3. The first test setup allows a go/no go check on the UART and support circuits. Connect START and STOP to the keyboard scan latch (explained in Figure 9). Connect CLOCK to $\overline{RD}$. Take signatures on data bus lines. If bus signatures are correct, then the UART and decoder are OK. If incorrect, then take signatures on the decoder outputs. If decoder signatures are incorrect, suspect the decoder. If correct, then set up the second test.

4. The second test setup allows verification of a UART chip failure. Connect START to the serial output line, TSO, which will open the GATE on the first serial output bit (the start bit). Connect STOP to the TBE output, which indicates the end of a serial output word. Connect CLOCK to the UART clock, pin TCP. Take signatures on the serial side of the UART for node-level fault isolation.




```
HEX
ADRS   CONTENTS   LABEL   INSTRUCTION

C37F              SEE:                        ;SERIAL RS-232 PORT
C37F   AF                 XOR   A
C380   D3FE               OUT   (0FEH),A      ;CLOSE S.A. GATE
C382   3C                 INC   A
C383   D3FE               OUT   (0FEH),A      ;OPEN S.A. GATE
C385   3EC1               LD    A,0C1H        ;1200 BAUD, RS-232 PORT
C387   D3FE               OUT   (0FEH),A
C389   3E0F               LD    A,0FH         ;8 BITS/CHAR, 2 STOP BITS,
C38B   D3FD               OUT   (0FDH),A      ;EVEN PARITY.
C38D   3EAA               LD    A,0AAH
C38F   57                 LD    D,A
C390              TWICE:
C390   7A                 LD    A,D
C391   D3FC               OUT   (0FCH),A
C393   01DC05             LD    BC,05DCH      ;10 MS DELAY CONSTANT
C396              WAIT:
C396   0D                 DEC   C
C397   C296C3             JP    NZ,WAIT       ;21010 T-STATES LATER
C39A   05                 DEC   B
C39B   C296C3             JP    NZ,WAIT
C39E   DBFD               IN    A,(0FDH)
C3A0   DBFC               IN    A,(0FCH)
C3A2   DBFD               IN    A,(0FDH)
C3A4   7A                 LD    A,D
C3A5   FE55               CP    55H
C3A7   CA7FC3             JP    Z,SEE
C3AA   1655               LD    D,55H
C3AC   C390C3             JP    TWICE
```

Figure 20. The serial inputs and outputs of a UART are stimulated by this program along with the control and status registers. The UART is first set to send and receive serial words with eight data bits, even parity and two stop bits at 1200 baud. Next, the program writes the word AAH into the UART and waits for its transmission to complete. Then the status word is read onto the data bus along with the serial word received, and then the status word again. Finally 55H is loaded for transmission similar to the word AAH. The program jumps back to the beginning of the loop upon completion.

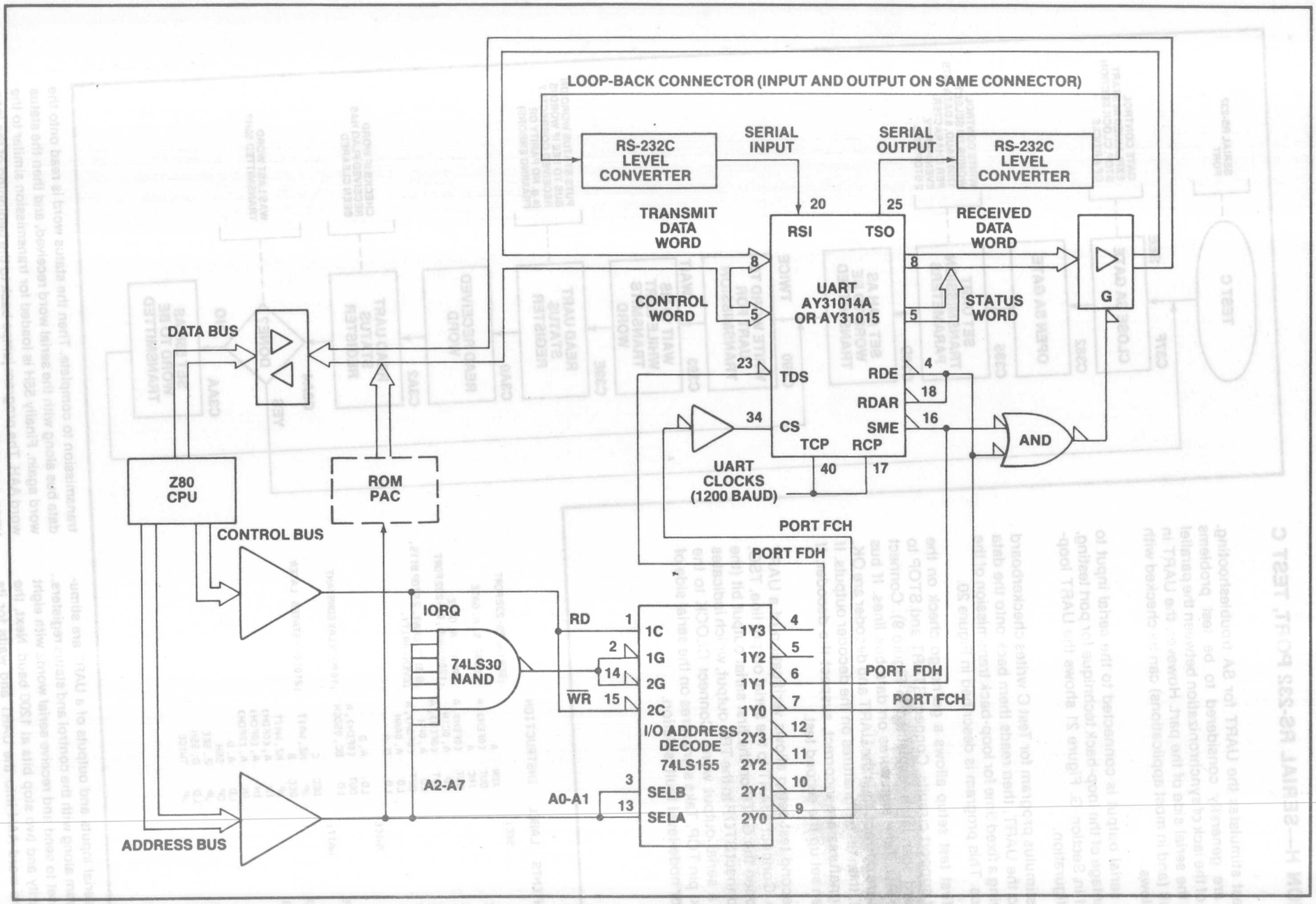


Figure 21. In Test C, the UART serial output is looped back into the serial input, as shown in this schematic. The stimulus exercises the entire loop. START, STOP and CLOCK connections for trouble-shooting both the parallel and serial sides of the UART are outlined in the text.

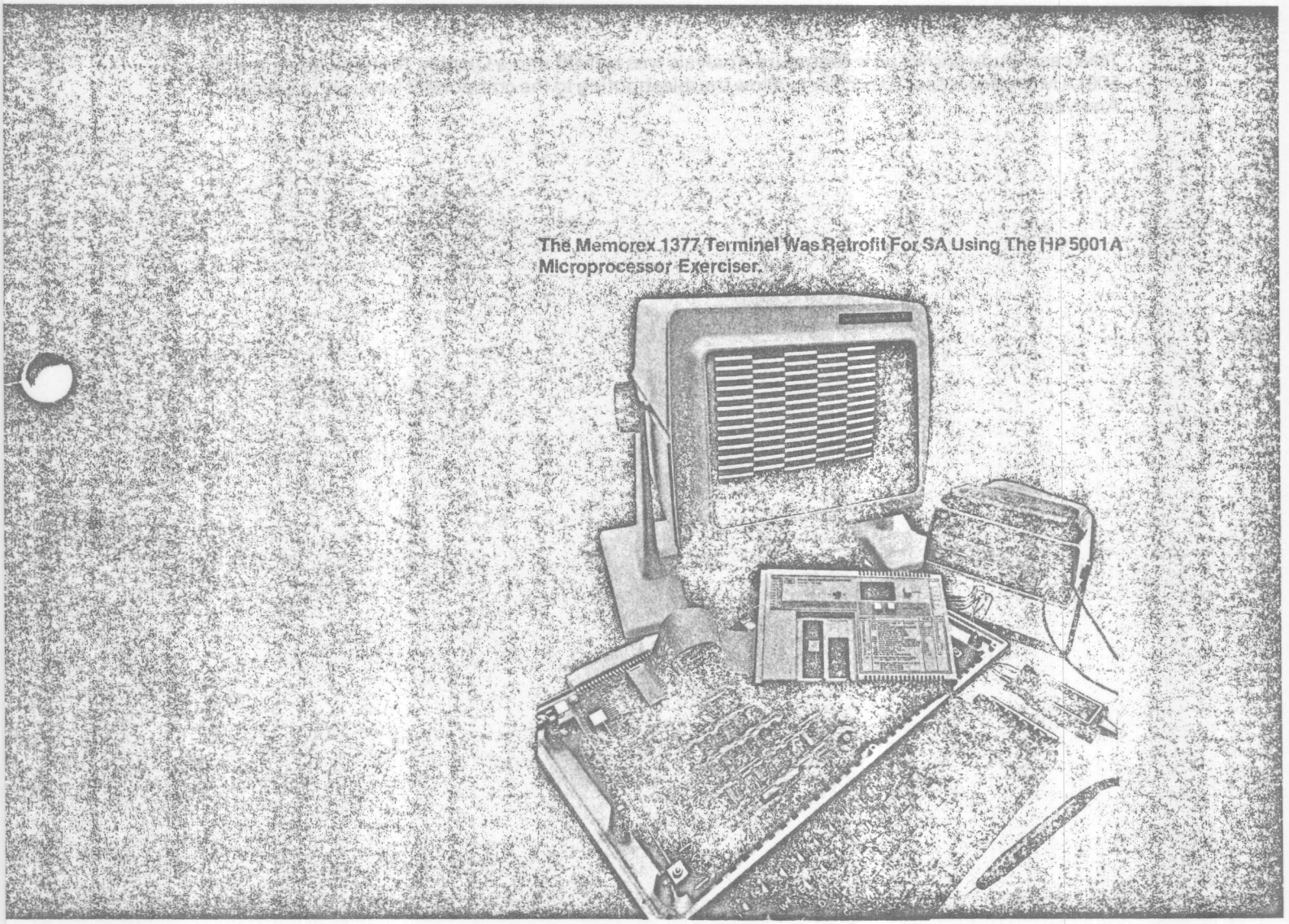
SECTION I—A DESIGNER'S CHECKLIST FOR GETTING STARTED WITH SA

Here's a summary of this article that can be used as a checklist when designing or retrofitting SA into a microprocessor based system. It is not limited to Z80-based systems or personal computers.

1. Provide a means to FREERUN the microprocessor by using a FREERUN fixture or by designing in a way to open the data bus and force the NOP or FREERUN instruction into the processor. Find START, STOP and CLOCK connections on the processor.
2. Provide a means to store, access, and select the SA stimulus programs. Try to find a way that depends only upon circuits that can be troubleshot with FREERUN or logic probes in a simple manner.
3. Create START and STOP using software to control hardware that's already available or hardware that is specially designed into the product for SA testing. Use circuits that can be checked by FREERUN, a previous SA test, or other easy means such as logic probes.
4. Choose a CLOCK that is synchronous to START, STOP and DATA on all nodes being tested. Be sure there's a CLOCK edge both before and after the START and STOP edges. Avoid CLOCKing DATA from a node when it's in the 3rd state.
5. Create software test loops that can give both a go/no-go indication of all bused devices (like a diagnostic) and also allow fault isolation of a bad component or process fault even with bused devices. The tests can be separate.
6. Be sure your test can isolate a failure because of either a read or write problem with the device (e.g. RAM).
7. Provide a way to stimulate uncontrolled inputs of I/O devices in a synchronous fashion using loop-back connectors or external stimuli.
8. Provide means to open feedback loops. Usually only a concern in circuits that are independent of the processor.
9. One-shots and UART serial outputs generally cannot be tested with SA, so find a way to bypass them (eliminate their effect on other circuit elements) during the SA test so that all nodes operate synchronous to the CLOCK.
10. Be sure your tests don't depend upon circuits working that are being tested. Usually done by running the SA tests open-loop (i.e. the tests only stimulate the devices but don't check the results to see if it was accomplished. The signature analyzer will check the results.). Sometimes can happen inadvertently when controlling START and STOP with software.

4

Application Note 222-11
A SIGNATURE ANALYSIS CASE STUDY
of a 6800-Based Display Terminal



The Memorex 1377 Terminal Was Retrofit For SA Using The HP 5001A Microprocessor Exerciser.

HEWLETT PACKARD

Application Note 222-11

A SIGNATURE ANALYSIS CASE STUDY of a 6800-Based Display Terminal

This case study shows how Signature Analysis was retrofit into a Display Terminal using the HP 5001A Microprocessor Exerciser to allow troubleshooting to the component level with a Signature Analyzer.

FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to bulid, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

ABOUT THIS CASE STUDY SERIES

Use of a Signature Analyzer requires that some test features be designed or retrofit into the product to be tested. This application note is one in a series of case studies aimed at assisting designers, test engineers, and others in understanding these features so that they can easily add Signature Analysis to their product. These case studies show detailed examples of these features in various digital systems based on specific microprocessors.

ABOUT THIS PUBLICATION

This is a reprint of a technical article from *Electronics* magazine. It describes how Signature Analysis testing was implemented on the 6800 microprocessor based Memorex 1377 Display Terminal. The terminal had not originally been designed with SA in mind, yet had been in production for several years. A new tool, the HP 5001A Microprocessor Exerciser, allowed SA to be easily retrofit into this existing product for faster testing and troubleshooting of the terminal on the manufacturer's production line. This article shows how the preprogrammed tests of the 5001A were selected and used to test the 6800, address and data buses, program ROM, scratchpad RAM, display RAM, and clock and timing circuits. Also included is a description of the custom program for testing the display refresh circuits in both a functional and diagnostic mode. Appended to the article is a listing of the custom program written to test the PIAs. The 5001A, combined with a special tool built by Memorex (circuit diagram shown), allowed testing of the asynchronous communications interface (IBM 3270 protocol). Results of circuit coverage and test effectiveness are covered in the article.

ABOUT OTHER PUBLILCATIONS

Application Note 222-0, "An Index to Signature Analysis Publications" lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests, from how to design or retrofit Signature Analysis into digital systems, to the cost reductions that can be expected in production test and field service by doing so. It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

CONTENTS

Applying signature analysis to existing processor-based products

With the advent of an off-board source of stimuli for signature analysis, it is possible to retrofit this efficent test method without extensive redesign

by Robert Rhodes-Burke,* *Hewlett-Packard Co., Santa Clara, Calif.*

☐ The digital test technique known as signature analysis (SA) is now widely regarded as the best way to service digital, processor-based boards. Until recently, however, products had to be designed to meet the special requirements of SA. In particular, the digital test patterns needed to stimulate the board during SA had to be generated by an on-board processor working from a user-supplied program.

Consequently, the many processor-based products designed before the technique's introduction could not be tested by it unless they underwent expensive redesign. However, now that an inexpensive, general-purpose stimulus source—the model 5001 microprocessor exerciser—is available, existing products can be serviced with little or no modification using signature analysis.

The application of SA to the Memorex 1377 display terminal will demonstrate the ease with which the technique can be used on equipment not designed for it. Although the terminal has been in production for several years and was not originally designed for signature analysis, it is a good SA target. Large portions of its circuity can be accessed through its processor using the 5001, so that a reasonably thorough check of its operation can be performed.

Understanding the application of SA to the 1377 demands familiarity with both the technique (see "Signature analysis revisited," p. 128) and the tools of the process. A brief description of the 5001 exerciser, its capability, and general application therefore precedes an explanation of the 1377 retrofit.

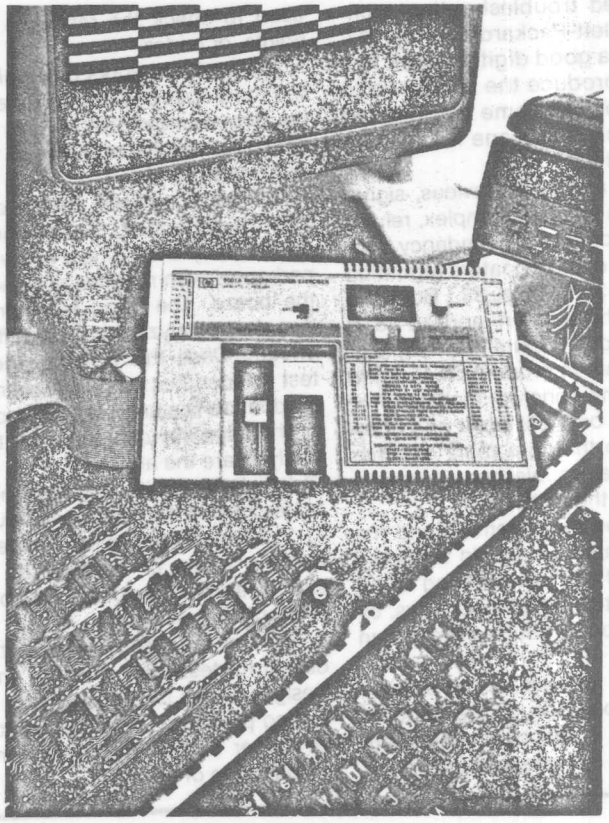
The model 5001 microprocessor exerciser, or Stimpod, as it is nicknamed, is a companion to the signature analyzer. It provides the digital stimuli to the board from which the analyzer takes bit streams for translation into hexadecimal signatures. Weighing only 3 pounds and measuring 9¼ by 5½ by 1 inch, the Stimpod is portable and travels easily into the field (Fig. 1).

Each Stimpod is designed for use with a particular type of microprocessor. The 6800 is supported at present, and the 8080, 8085, and Z80 will be supported in the coming year.

To employ the Stimpod, the processor on the board under test is removed and plugged into a zero-insertion-force connector on the front panel of the 5001. A ribbon cable from the Stimpod is then connected to the board socket that has been vacated by the processor.

The processor and board can then be exercised by the 5001. Contained in the Stimpod's internal memory are 52 test programs (see Table 1) applicable to a general class of microprocessor-based designs. For testing special aspects of a design, another ZIF socket is provided on the 5001's front panel. It accepts a 2716 erasable programmable read-only memory in which the user has placed a stimuli program. Thus the 5001 lets users apply signature analysis techniques without designing in or disturbing on-board program memory.

To use one of the internal test programs, a user sets the front-panel switch to



1. Retrofit in a box. Using signature analyzers to troubleshoot products not designed for use with them like the Memorex 1377 terminal is easy, thanks to the 5001 microprocessor exerciser (center). The checkerboard pattern (top) is produced by a 5001 program.

*The author is now with Apple Computer Corp., Cupertino, Calif.

INT and calls up the test by number, pressing the tens and units buttons below the light-emitting-diode display. Pressing the enter button then starts the test. Putting a preprogrammed PROM in the smaller front-panel socket and moving the double-throw switch to the external position permits custom tests to be called up and run in the same manner.

Since the 5001 was designed for use with a signature analyzer, it produces the start, stop, and clock signals needed by that unit to gate the bit stream and form a signature. These signals are provided at ports on the right of the exerciser, along with a common ground and a qualifier signal that is applicable in some of the preprogrammed tests.

On the left side of the Stimpod is an eight-bit output port, a qualifier input, and a power input and ground. Using the byte-wide port, the 5001 can stimulate the input side of input/output devices on the product being tested. The qualifier line can be used to recognize when a particular device is enabled, so that the 5001 can tell the signature analyzer to collect bits going to or coming from the device. The external power ports need only be used when the exerciser's power requirements—nominally 2.75 watts, exclusive of the processor—exceed the power available directly from the board under test.

Signature analysis could be applied to most of the 1377 by the 5001 and a 5004 signature analyzer alone. The general methodology of SA application involves deriving a set of signatures for a known good board, analyzing possible faults to determine how they change the signatures, and verifying that the set of signatures thus generated is valid and unique using other boards of the same type.

Although all this was done by hand for the 1377, it can now be almost completely automated with a automatic board tester that incorporates signature analysis. Such a board tester is available from Hewlett-Packard—the 3060A with option 100. In either case, once valid signatures are obtained, the board designer or test engineer can generate a fault tree—a listing or schematic that tells what points to check next if a fault signature is found. A field technician otherwise ignorant of SA can then use this tree to troubleshoot a system.

Applying signature analysis to many microcomputer-based products is often straightforward, using only the preprogrammed tests in the 5001. The 1377, however, is a special challenge to SA application because of some unusual aspects of its design. These can be understood by examining the functional layout of the terminal.

Choice target

The digital electronics used in the model 1377 display terminal are contained on a single, 9-by-16-in. multilayered printed-circuit board (Fig. 2). The board can be functionally segmented into five separate areas: the microcomputer, the display memory, the display-refresh algorithmic state machine (ASM), the communications ASM, and the clock and timing circuitry.

It should be noted that in normal operation three machines—the microcomputer, the communications

Signature analysis revisited

Signature analysis, a patented troubleshooting technique introduced by the Hewlett-Packard Co. in 1977, is based on the principle that a good digital circuit in a known (initialized) state will produce the same output when stimulated repeatedly by the same input. If the repeated output of a device is not the one it has been designed to produce, it has failed.

While this principle is simple and fairly obvious, signature analysis implementation is a bit more complex, relying on mathematics similar to that for cyclic redundancy coding. (For a thorough explanation of the signature formation process, see *Electronics*, March 3, 1977, p. 93.) But once signature analysis techniques have been applied to a design, using them to troubleshoot it is extremely simple.

All a technician has to do, either in production or in the field or depot, is follow the time-honored technique of signal tracing. He or she checks one test point and compares the measurement result to that in a table or schematic. If it does not match, the troubleshooter checks another point in accordance with the test plan.

The point between the last bad measurement and the next good one is the failure location. It should be noted that the technician does not need to know anything about signature analysis or, for that matter, digital logic. Thus, by requiring less from the troubleshooter, the technique can greatly reduce service costs.

In the past, a repetitive digital stimulus, or bit stream, was generated by the microprocessor of the product under test, which would execute a special test program residing in an on-board memory. A signature analyzer is used to check the response at the board's various test points, or nodes.

The analyzer works by monitoring the bit stream on one line for a specified period determined by the clock rate of the circuit under test. In this process it compresses the data and translates it into a four-character hexadecimal word, or signature.

A set of signatures, one for each test node in the circuit, must be generated; the designer or test engineer generally does this by exercising a known good unit—the prototype perhaps—with routines designed to exercise each area of the board. The good signatures are recorded; then the board is analyzed for the effect of failures on the bit stream. Once this is done, the test designer can create test procedures and test-point schematics for use by production-line and field technicians.

The three key conditions for the application of signature analysis are the ability to:

- ☐ Initialize circuits that can hold two or more different states (RAMs, flip-flops, and counters, for example).
- ☐ Synthesize the timing needed for signature sampling, both in terms of framing the sequence (sample start-sample stop) and clocking the bit stream.
- ☐ Apply the stimulus.

The first of these requirements is really a basic tenet of design for testability and can only be achieved by design. Designers who fail to observe it will be faced with a major testing problem. The other two conditions, however, need no longer be absolute design imperatives, thanks to the introduction of the 5001. **-Richard W. Comerford**

TABLE 1: MICROPROCESSOR EXERCISER TEST SET

Number	Test	Address range	Qualifier
00	Microprocessor: 6800 instruction set, interrupts	—	—
01	Buses: free-run	all	—
02	RAM: read/write 6800 direct addressing range	0000 – 00FF	—
03	RAM: read/write multiple patterns (checker-	0000 – 3FFF	—
04	board, inverse, address-as-data);	4000 – 7FFF	—
05	address range is selected according	8000 – BFFF	—
06	to test number	C000 – FFFF	—
07	RAM: read/write address as data	all	—
08	RAM: read/write alternating checkerboard	all	—
09	RAM: write checkerboard, then free-run	all	—
10/11	I/O: write patterns to qualified outputs	as qualified	0/1
12/13	I/O: read stimulus from qualified inputs	as qualified	0/1
14/15	ROM: read qualified data	as qualified	0/1
16/17	ROM: bus signature (pin X_0)	as qualified	0/1
18,19	5001: self-exercise	—	—
20 – 51	ROM: read 2-K address range	*	—

*Test number indicates 2-K address range, for example, 20 is 0000 to 07FF, 51 is F800 to FFFF

ASM, and the display refresh ASM—are jointly responsible for the terminal's workings. They operate concurrently in a complex, interleaved fashion, using various bus arbitration methods to share the data and address lines and access the shared display memory. The challenge, then, was to find a way of checking the functional areas independently, yet as they really performed.

Other aspects of the 1377's design were challenging, too. The use of two programmable interface adapters (PIAs) in the microcomputer section required a somewhat custom approach, since the designation of these devices' ports for input or output is determined and programmed to fit a specific application. In the 1377, the PIAs scan the terminal's keyboard and configuration switches, as well as drive its audio output (bell) and status indicator. Data from the PIAs can directly control some of the functions of the ASMs, such as selecting different segments of the display RAM or inhibiting the display or its reset. Therefore, it was essential to find a way to check the operation thoroughly.

The most elaborate part of the board is the timing and clock circuitry. Like other clock circuits, it generates timing waveforms independently of the microprocessor. On the 1377, it generates 11 phased clocks—more than usual—which are in turn used by the other functional blocks for timing generation. Despite this circuitry's complexity, SA checked it 100% with the least difficulty.

Of the other four functional blocks, the microcomputer also was 100% tested with SA. But the display memory block could be only 90% tested and the display-refresh ASM could be only 80% tested because of the communications ASM.

The communications ASM operates asynchronously with respect to the rest of the system, so that it could not be checked using the 5001 and the 5004 alone. However, a special test tool designed previously by Memorex had been used for some time to check this portion of the terminal. With some adaptation of such a tool (to be described later), even this asynchronous segment could be tested with signature analysis techniques.

The microcomputer section was (and in general is) the logical place to begin applying signature analysis. It is the area in which the 5001 exercises most direct control over the system operation. Further, the elements of the block are extremely common and hence can be easily checked out with the 5001's preprogrammed tests.

The 5001 was connected to the 1377 as described earlier, and the 5004's start, stop, and clock inputs were connected to the Stimpod. The 5004 was set to operate from the rising edge of the clock and start signals and the falling edge of the stop signal. For most of the tests performed, this was the only setup needed.

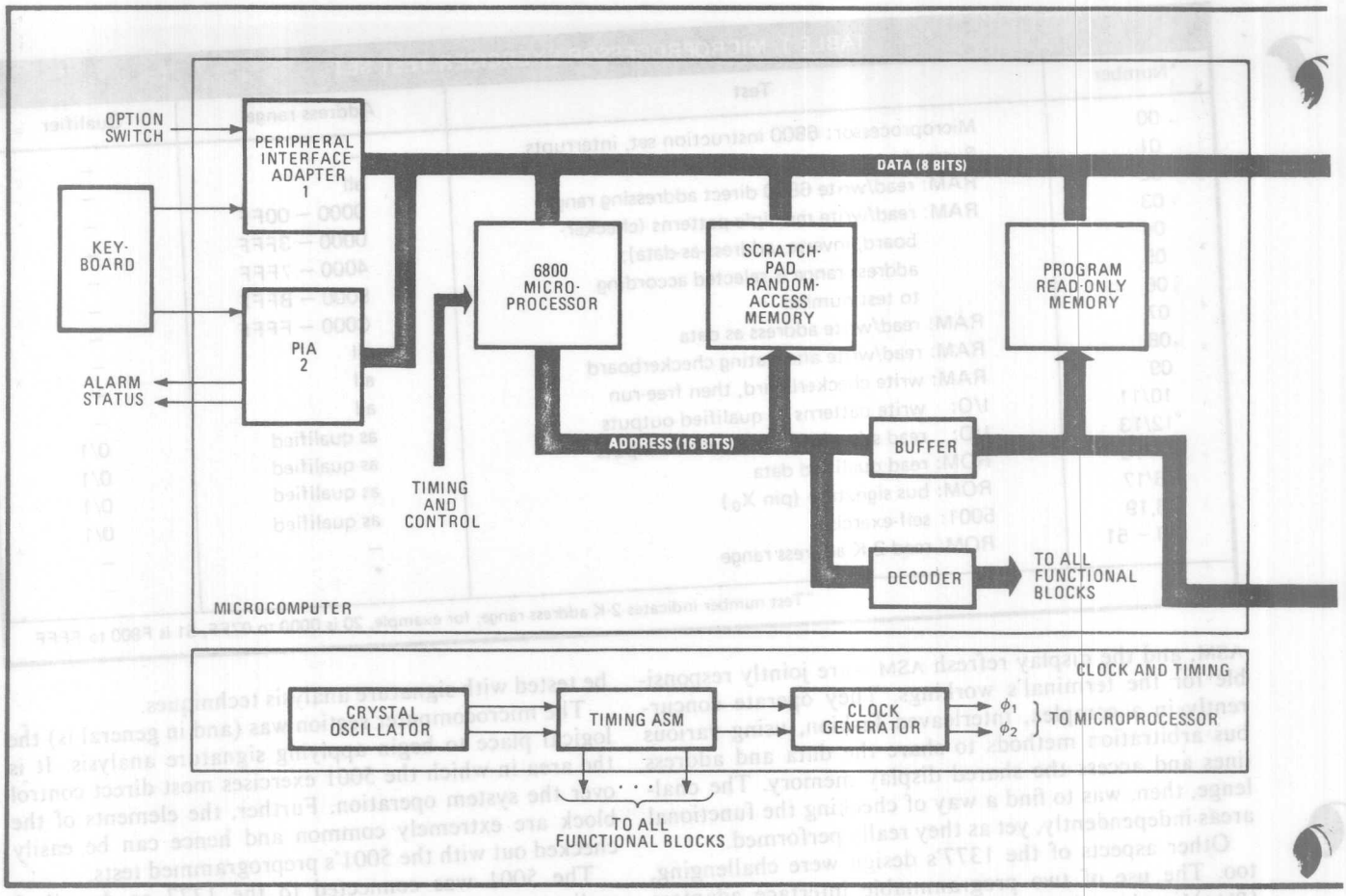
Test 00 (see Table 1) was the first performed. It checks the 6800's functionality by running it through its entire instruction set, except the wait-for-interrupt instruction, while it is isolated from the board. Other instructions in the test sequence serve to verify the interrupt line operation and the processor's ability to service interrupts.

Checking out the kernel

While the test is running, the functioning of the kernel—the processor, its clock and power—can be verified by taking a single signature. If the kernel is operating, holding the 5004's data probe to the processor's +5-volt supply produces a bit stream of 1s, which results in a signature of $28PH_{16}$. This provides an 80% confidence level that the processor is good.

To provide an even higher confidence level (95%), each pin of the 6800 can be probed while the test is run. This will result in the signatures shown in Fig. 3 and takes about 30 seconds to perform. Unless other tests indicate that the processor could be at fault, however, this check is not absolutely necessary.

Once the kernel's operation is verified, the next step is to check that the address lines and decoders are operational so that the Stimpod can access other circuits for test. Another preprogrammed test, 01, is used to do this



2. Shared facilities. In the 1377, the microprocessor, display refresh ASM, and the communications ASM share buses and the display memory. A major challenge is finding a way to independently analyze how these blocks work through the shared facilities.

check. In this free-run test, the CPU runs through its entire address repertoire, the analyzer's data probe is placed on each of the address lines, and a signature is obtained for each (Table 2). These signatures will be the same for any 6800 processor tested in this way.

Using the same test, signatures can be obtained at the outputs of the address decoders, but since their output is product-dependent, the signatures will vary from one design to another. After the ability of the processor, and thus the Stimpod, to access the various components of the system has been verified, they can be checked. The most logical area to investigate next is one closest to the processor, such as the memory in which the operating program is contained.

One-signature checking

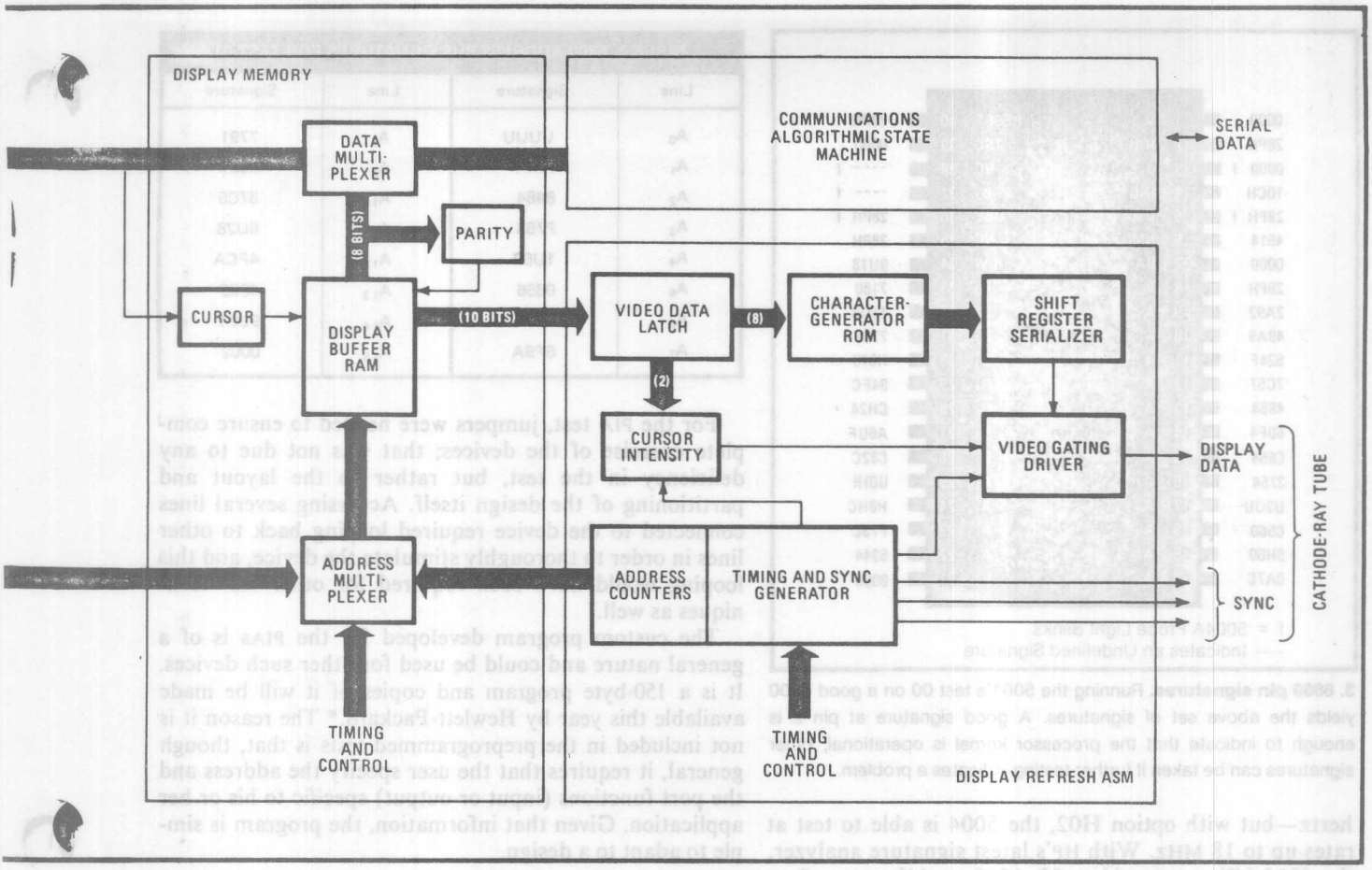
Depending on the product design, the program memory—read-only memory, or ROM—can be checked with a single signature or one signature for each memory device in it. This is done using either preprogrammed test 16 or 17, depending on whether the memory-enable line is active high or low, respectively.

In both tests, the entire contents of the enabled memory are read by the 5001 and formed into a single bit stream, which is output through port X_0 on the Stimpod's left side. By placing the data probe of the signature analyzer on this port, a signature that is unique to the content of the memory being tested can be taken.

Up to three ROMs contain the operating program in the 1377. To check each individually, the 5001's Q-in is tied to a chip-enable line and the contents of the chip are read by the Stimpod; test 16 does this since the 1377's chip-enable lines are active low. Were a block-enable line used in the design, the entire chip set could be checked with a single signature. Then, if an incorrect signature resulted, the contents of each chip could be checked individually.

The 128-byte scratchpad random-access memory was checked next, using test 02, read-write 6800 direct-addressing range. This test uses a sliding-1 pattern with a reverse background fill, which detects address interdependency. After the Stimpod writes the pattern, it reads it and the values written and read determine the clocking signal it supplies to the signature analyzer. Thus, by placing the signature analyzer on the +5-V supply line as in the microprocessor test, a signature is produced that reflects the number of good cells in the RAM. RAMs with less than 256 bytes can be checked using this test; other tests are provided for larger RAMs.

In the 1377, PIA_2 falls within the direct address range of the 6800. Thus, the PIA was partially exercised using test 02, when it was responding as though there were two extra scratchpad RAM cells when it was operating properly. If an incorrect signature appeared, the PIA-enable



line was grounded and the test rerun to see if the fault was actually in the RAM or the PIA.

With the completion of the scratchpad RAM test, the entire microcomputer block with the exception of the PIAs was completely checked. To test the PIAs, it was determined that a custom program would be needed. Rather than change the test setup and insert a custom programmable ROM, it was decided to save time and continue testing whatever could be checked with the preprogrammed tests and the present setup. The custom PIA test will be described later, along with a discussion of other custom tests.

The display memory, or display buffer RAM, in the 1377 is a 4,098-by-10-bit store composed of 10 1-K RAMs. This is divided into two pages, each consisting of 2,049 10-bit words. As the microprocessor can handle only bytewide data, two accesses (and therefore two addresses) are needed to obtain the data set; one address maps to the cursor portion of the stored data, while the other maps to the character portion.

Since the display memory address range is therefore large, two preprogrammed tests were first used to check its operation: test 03, which covers the address range from 0000_{16} to $3FFF_{16}$, and test 04, from 4000_{16} to $7FFF_{16}$. Before these tests were run, the display refresh was disabled by grounding a single pin, in order to prevent it from interfering with the 5001's control of the display RAM.

As shown in Table 1, these tests cause a number of different patterns to be read from and written to the RAM, thoroughly checking its operation. While the tests are run, the analyzer's data probe is set on each of the data-in and data-out pins of the two-port RAMs. The test resulted in only 20 signatures being generated for the entire display RAM.

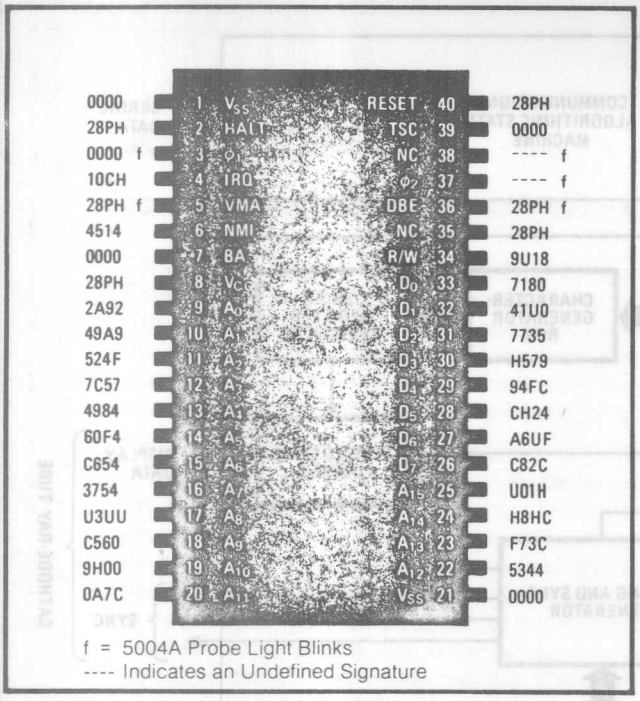
Though the preprogrammed routines were well suited to testing the display memory, it was subsequently found that a custom program needed to test the display refresh circuitry could also check display-memory operation, and was more effective in that it resulted in the need for even fewer signatures. This test will be described later, together with other custom programs.

Clock and timing

As previously noted, the clock and timing circuitry of the 1377 terminal is elaborate and takes up a considerable portion of the board's electronics. Even so, it was a relatively straightforward matter to derive a set of signatures for it.

Since the circuit is self-stimulating, it runs asynchronously and independently of the processor. For this reason, the start, stop, and clock inputs to the 5004 were moved from the Stimpod to the circuit itself; it was not necessary to reset the edges for these inputs.

Carrying out the tests at the clock rate of the circuit was very desirable; taking signatures in this way is simpler and the results are more easily interpreted. The 1377's clock rate is very high—greater than 17 mega-



3. 6800 pin signatures. Running the 5001's test 00 on a good 6800 yields the above set of signatures. A good signature at pin 2 is enough to indicate that the processor kernel is operational; other signatures can be taken if further testing indicates a problem.

hertz—but with option H02, the 5004 is able to test at rates up to 18 MHz. With HP's latest signature analyzer, the 5005 [*Electronics*, Nov. 20, 1980, p. 44], rates of up to 20 MHz can now be checked.

Before checking the circuit, two preprogrammed tests were run on the Stimpod. Test 09 was run first to fill the display RAM with a checkerboard pattern; this predictable pattern allowed some stable signatures to be taken in the area of the character generator. Before any signatures were taken, however, test 32 was run. This test tells the processor to read a 2-K address range that does not affect the shared buses and thus limits its activity to an area removed from the one to be checked. Thus, timing circuits outside the actual clock generator itself could also be investigated.

The 1377's clock and timing circuit is a long, chain-like circuit with a fair amount of feedback. It was therefore necessary to move outside the loop to verify correct inputs. Though it may be necessary to break a loop when extensive feedback exists, it is not absolutely required. When provision is made for resetting the counter chain, as it was in the 1377, it is often not necessary to break it, particularly if the loop contains only two or three elements. Isolating a fault to such a loop is then sufficient, permitting the faulty element to be easily found with the signature analyzer's built-in logic probe.

As noted previously, two areas of the board required some custom approaches to generating signatures: the PIAs and the display-refresh circuitry. Both tests involved programming a 2716 erasable PROM, but this was the only thing necessary for testing the display-refresh ASM.

*See Appendix A

TABLE 2: 6800 ADDRESS BUS SIGNATURES FOR TEST 01

Line	Signature	Line	Signature
A_0	UUUU	A_8	7791
A_1	FFFF	A_9	6321
A_2	8484	A_{10}	37C5
A_3	P763	A_{11}	6U28
A_4	1U5P	A_{12}	4FCA
A_5	0356	A_{13}	4868
A_6	U759	A_{14}	9UP1
A_7	6F9A	A_{15}	0002

For the PIA test, jumpers were needed to ensure complete exercise of the devices; that was not due to any deficiency in the test, but rather to the layout and partitioning of the design itself. Accessing several lines connected to the device required looping back to other lines in order to thoroughly stimulate the device, and this looping would have been required for other test techniques as well.

The custom program developed for the PIAs is of a general nature and could be used for other such devices. It is a 150-byte program and copies of it will be made available this year by Hewlett-Packard.* The reason it is not included in the preprogrammed tests is that, though general, it requires that the user specify the address and the port functions (input or output) specific to his or her application. Given that information, the program is simple to adapt to a design.

Testing display-refresh

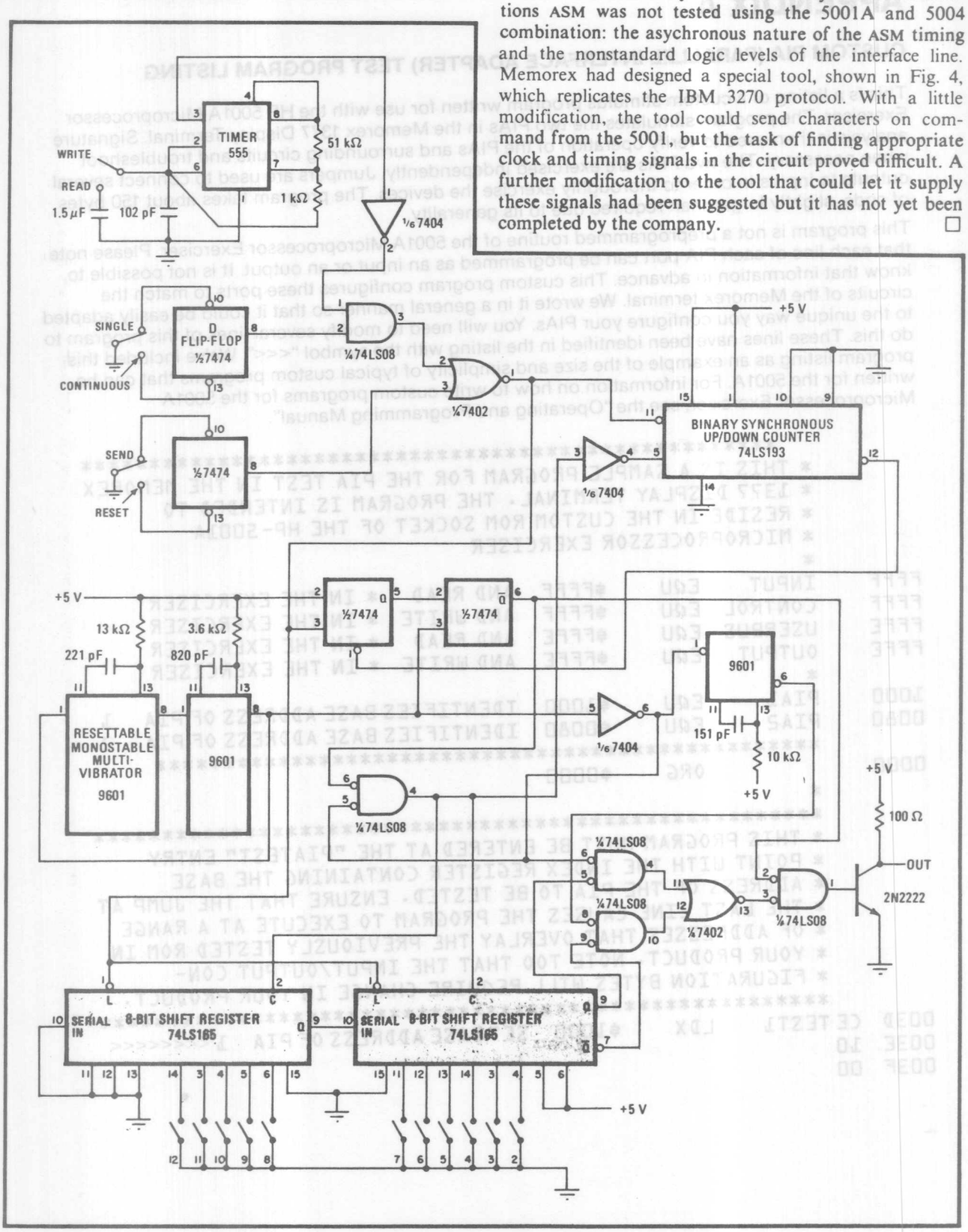
Two levels of display-refresh testing—functional and diagnostic—were provided for the 1377 with one custom program. This program initialized the display RAM with all possible display characters and some nondisplayed control sequences. It was, in effect, similar to the "quick brown fox . . ." type of test often used in data communications testing.

For both tests, the framing and clock signals input to the signature analyzer were taken from the display-refresh ASM, since it runs asynchronously with respect to the microprocessor. The clock signal for the functional test was taken from the dot-clock that controls the video gating drivers. Using this clock, the analyzer can verify that about 40% of the terminal's circuits are functional with only one signature, which is taken by placing the analyzer's data probe on the display-refresh ASM's video output. This quick and easy go/no-go test also provides a full CRT display.

It should be noted that with this custom program both the microcomputer and display-refresh ASM are running concurrently, just as they would in normal interleaved operation. For the diagnostic check of the display-refresh ASM, the signature analyzer's clock is moved to the character clock, which is synchronous with direct memory accesses performed by that ASM. Thus, by using this clock to check activity on the system buses, signatures are generated only when the display-refresh is using them—the interleaved operation is separated for testing.

Both the functional and diagnostic approaches used in

4. Serial stimuli. For checking out the serial interface (the communications ASM) of the 1377, Memorex has been using the in-house tester, which simulates IBM 3270 protocol. Setting toggle switches causes two 8-bit registers (shaded) to create a test word.



the 1377 could easily be applied to other products. They may, in fact, prove invaluable in testing the increasing number of processor-based CRT products.

There were basically two reasons why the communications ASM was not tested using the 5001A and 5004 combination: the asychronous nature of the ASM timing and the nonstandard logic levels of the interface line. Memorex had designed a special tool, shown in Fig. 4, which replicates the IBM 3270 protocol. With a little modification, the tool could send characters on command from the 5001, but the task of finding appropriate clock and timing signals in the circuit proved difficult. A further modification to the tool that could let it supply these signals had been suggested but it has not yet been completed by the company. □

APPENDIX A

CUSTOM PIA (PARALLEL INTERFACE ADAPTER) TEST PROGRAM LISTING

This is a listing of a custom stimulus program written for use with the HP 5001A Microprocessor Exerciser. The program stimulates the two PIAs in the Memorex 1377 Display Terminal. Signature analysis is then used to verify operation of the PIAs and surrounding circuits and troubleshoot when necessary. The two PIAs are exercised independently. Jumpers are used to connect several outputs to inputs in order to thoroughly exercise the devices. The program takes about 150 bytes of code, slightly larger than required due to its generality.

This program is not a preprogrammed routine of the 5001A Microprocessor Exerciser. Please note that each line of each PIA port can be programmed as an input or an output. It is not possible to know that information in advance. This custom program configures these ports to match the circuits of the Memorex terminal. We wrote it in a general manner so that it could be easily adapted to the unique way you configure your PIAs. You will need to modify several lines of this program to do this. These lines have been identified in the listing with the symbol "<<<". We've included this program listing as an example of the size and simplicity of typical custom programs that can be written for the 5001A. For information on how to write custom programs for the 5001A Microprocessor Exerciser, see the "Operating and Programming Manual".

```
                ****************************************************
                * THIS IS A SAMPLE PROGRAM FOR THE PIA TEST IN THE MEMOREX
                * 1377 DISPLAY TERMINAL. THE PROGRAM IS INTENDED TO
                * RESIDE IN THE CUSTOM ROM SOCKET OF THE HP-5001A
                * MICROPROCESSOR EXERCISER
                *
FFFF    INPUT   EQU     $FFFF   AND READ    * IN THE EXERCISER
FFFF    CONTROL EQU     $FFFF   AND WRITE   * IN THE EXERCISER
FFFE    USERBUS EQU     $FFFE   AND READ    * IN THE EXERCISER
FFFE    OUTPUT  EQU     $FFFE   AND WRITE   * IN THE EXERCISER
                *
1000    PIA1    EQU     $1000   IDENTIFIES BASE ADDRESS OF PIA  1
0080    PIA2    EQU     $0080   IDENTIFIES BASE ADDRESS OF PIA  2
                ************************************************
0000            ORG     $0000
                *
                ****************************************************
                * THIS PROGRAM MUST BE ENTERED AT THE "PIATEST" ENTRY
                * POINT WITH THE INDEX REGISTER CONTAINING THE BASE
                * ADDRESS OF THE PIA TO BE TESTED. ENSURE THAT THE JUMP AT
                * THE LAST LINE CAUSES THE PROGRAM TO EXECUTE AT A RANGE
                * OF ADDRESSES THAT OVERLAY THE PREVIOUSLY TESTED ROM IN
                * YOUR PRODUCT. NOTE TOO THAT THE INPUT/OUTPUT CON-
                * FIGURATION BYTES WILL REQUIRE CHANGE IN YOUR PRODUCT.
                ****************************************************
003D  CE TEST1  LDX     $1000   SET BASE ADDRESS OF PIA  1 <<<<<<<<<
003E  10
003F  00
```

```
0040 7E          JMP    PIATEST+$6800   GO TO TEST OF PIA <<<<<<<<
0041 68
0042 49
         ********************************************************
0043 CE TEST2    LDX      $0080  SET BASE ADDRESS OF PIA   2 <<<<<
0044 00
0045 80
0046 7E          JMP      PIATEST+$6800  GO TO TEST OF PIA <<<<<<<<
0047 68
0048 49
         ********************************************************
0049 86 PIATEST LDA  A     $1C    START=STOP=0, EMIT SA
                                    READ CLOCKS
004A 1C
004B B7          STA  A   CONTROL
004C FF
004D FF
004E 86          LDA  A     $1F    START=STOP=1, EMIT SA RD AND
                                      WR CLK
004F 1F
0050 B7          STA  A   CONTROL
0051 FF
0052 FF
0053 4F          CLR  A             CLEAR DDRA ACCESS BIT=0
0054 A7          STA  A     1,X     FOR A-SIDE OF PIA
0055 01
0056 86          LDA  A     $FF     SET DDR TO ALL OUTPUTS <<<<<<<<<<
0057 FF
0058 A7          STA  A     0,X     ON A-SIDE OF PIA
0059 00
005A 86          LDA  A     $04     SET    DDRA ACCESS BIT=1
005B 04
         *
         * NOW WE HAVE ACCESS TO DATA REGISTER ON THE A-SIDE
         *
005C A7          STA  A     1,X     FOR A-SIDE OF PIA
005D 01
005E 4F          CLR  A
005F A7 STIMA1   STA  A     0,X     WRITE PATTERN TO OUTPUTS
0060 00
0061 E6          LDA  B     0,X     READ BACK RESULTING LEVELS
0062 00
         * NOTE THAT THE A-SIDE IS UNBUFFERED, AND RESULTING DATA
         * IS THE SAME AS ACTUALLY SEEN BY THE OUTPUT OF THE PIA
         *
0063 4C          INC  A             INCREASE PATTERN BY ONE
0064 26          BNE      STIMA1    CONTINUE UNTIL 256 PATTERNS WRITTEN
0065 F9
0066 20          BRA      BSIDE
0067 02
0068 20 BACKPIA BRA      PIATEST
0069 DF
         * NOW THE B-SIDE
```

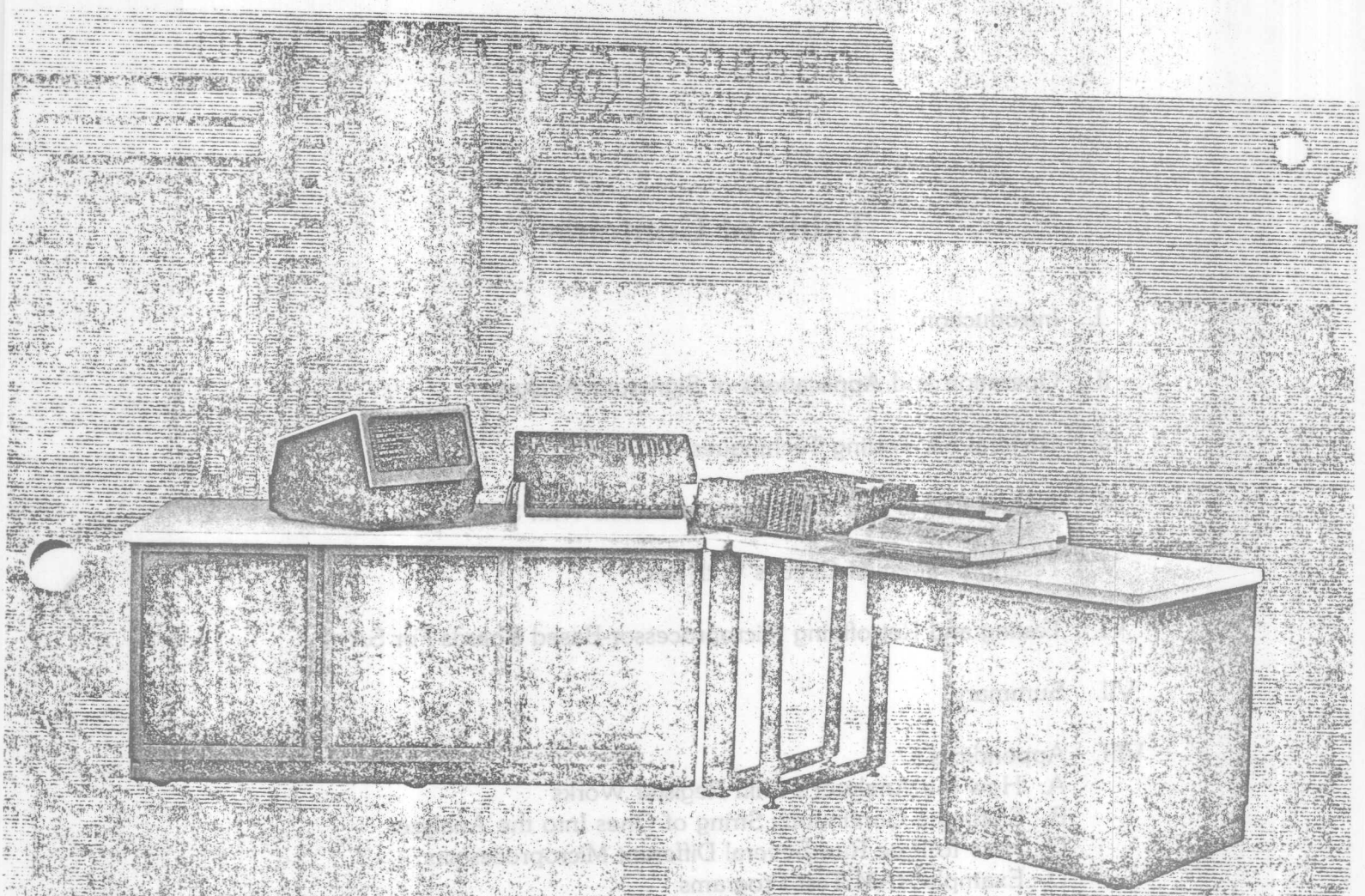
```
006A  86 BSIDE   LDA  A    $01   SET X0, CLEAR X1 BITS IN
006B  01
006C  B7         STA  A   OUTPUT  OUTPUT LATCH ON 5001A
006D  FF
006E  FE
006F  4F         CLR  A           CLEAR DDRB ACCESS BIT=0
0070  A7         STA  A    3,X    FOR PIA  B-SIDE
0071  03
0072  86         LDA  A    $7F    SET DATA DIRECTION 7 OUT 1 IN <<<
0073  7F
0074  A7         STA  A    2,X    FOR PIA    B-SIDE
0075  02
0076  86         LDA  A    $04    SET DDRB ACCESS BIT=1
0077  04
0078  A7         STA  A    3,X    FOR PIA    B-SIDE
0079  03
                 *
                 * NOW THE ACCESS BIT IS SET, WE CAN ACCESS THE DATA
                 * REGISTER
                 *
007A  4F         CLR  A           RESET PATTERN TO ZERO
007B  A7 STIMB1  STA  A    2,X    FOR PIA    B-SIDE
007C  02
007D  4C         INC  A           NEXT PATTERN
007E  26         BNE    STIMB1  UNTIL ALL 256 POSSIBLE ARE WRITTEN
007F  FB
                 *
                 * HERE WHEN ALL PATTERNS WRITTEN, READ THE INPUT PIN
                 *
0080  A6         LDA  A    2,X    FOR PIA    B-SIDE
0081  02
0082  C6         LDA  B    $03    GET INPUT BIT HIGH, LEAVE RESET HIGH
0083  03
0084  F7         STA  B   OUTPUT
0085  FF
0086  FE
0087  A6         LDA  A    2,X    FOR PIA    B-SIDE
0088  02
                 *
                 * NOW GO ON TO TEST CA1,CA2,CB1,CB2
0089  20         BRA     TESTCAB
008A  02
008B  20 PIABACK BRA     BACKPIA
008C  DB
                 *
                 * HERE WITH CA1 AND CA2 TIED TOGETHER IN THE KEYBOARD
                 * CONNECTOR, SINCE THE KEYBOARD IS NOT PRESENT.
                 *
008D  86 TESTCAB LDA  A    $34    SET CA2 LOW FIRST, NO EFFECT
008E  34
008F  A7         STA  A    1,X
0090  01
```

```
0091  4C          INC  A           NOW CONFIGURE CA1 FOR FALLING EDGE
0092  A7          STA  A    1,X
0093  01
0094  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
0095  01
0096  C6          LDA  B    #3D    FORCE A RISING EDGE ON CA1 VIA CA2
0097  3D
0098  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
0099  01
009A  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
009B  01
009C  C6          LDA  B    #35    FORCE A FALLING EDGE ON CA1 VIA CA2
009D  35
009E  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
009F  01
00A0  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00A1  01
00A2  C6          LDA  B    #36    PROGRAM FOR POS EDGE INTERRUPT ON IRQA
00A3  36
00A4  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
00A5  01
00A6  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00A7  01
00A8  C6          LDA  B    #3E    FORCE A RISING EDGE ON CA1 VIA CA2
00A9  3E
00AA  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
00AB  01
00AC  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00AD  01
00AE  C6          LDA  B    #36    FORCE A FALLING EDGE ON CA1 VIA CA2
00AF  36
00B0  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
00B1  01
00B2  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00B3  01
00B4  C6          LDA  B    #37    PROGRAM FOR NO PULL ON IRQA
00B5  37
      *                            IF INTERRUPT OCCURS
00B6  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
00B7  01
00B8  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00B9  01
00BA  C6          LDA  B    #3F    FORCE A RISING EDGE ON CA1 VIA CA2
00BB  3F
00BC  E7          STA  B    1,X    WRITE TO CONTROL REG A-SIDE
00BD  01
00BE  A6          LDA  A    1,X    READ STATUS ON A-SIDE OF PIA
00BF  01
```

```
00C0  C6          LDA  B     #$37   FORCE A FALLING EDGE ON CA1 VIA CA2
00C1  37
00C2  E7          STA  B     1,X    WRITE TO CONTROL REG A-SIDE
00C3  01
00C4  A6          LDA  A     1,X    READ STATUS ON A-SIDE OF PIA
00C5  01
                  *
                  * HERE WHEN DONE WITH SIDE A CA1 AND CA2
                  *
                  *
00C6  86          LDA  A     #$34   SET CB2 LOW FIRST, NO EFFECT
00C7  34
00C8  A7          STA  A     3,X
00C9  03
00CA  4C          INC  A            NOW CONFIGURE CA1 FOR FALLING EDGE
00CB  A7          STA  A     3,X
00CC  03
00CD  A6          LDA  A     3,X    READ STATUS ON B-SIDE OF PIA
00CE  03
00CF  C6          LDA  B     #$3D   FORCE A RISING EDGE ON CA1 VIA CB2
00D0  3D
00D1  E7          STA  B     3,X    WRITE TO CONTROL REG B-SIDE
00D2  03
00D3  A6          LDA  A     3,X    READ STATUS ON B-SIDE OF PIA
00D4  03
00D5  C6          LDA  B     #$35   FORCE A FALLING EDGE ON CA1 VIA CB2
00D6  35
00D7  E7          STA  B     3,X    WRITE TO CONTROL REG B-SIDE
00D8  03
00D9  A6          LDA  A     3,X    READ STATUS
00DA  03
00DB  4F          CLR  A            CLEAR DDRA ACCESS BIT=0
00DC  20          BRA       PIABACK REPEAT TEST
00DD  AD
```

Implementing Signature Analysis for Production Testing

with the HP 3060A Board Test System



Application Note 222-1

HEWLETT hp PACKARD

Table of Contents

I. Introduction

The advantages of production testing printed circuit boards with capabilities for functional and component level testing have been realized in recent years as board test systems and techniques have been refined. The testing of low complexity digital circuitry is well served on the HP 3060A Board Test System by a digital testing technique known as static pattern testing. As the digital circuitry's complexity increases (with feedback, multilevel logic, and increased IC count), the programming for this test capability becomes challenging and the time required to thoroughly test a board can sometimes make the cost effectiveness of this technique questionable. Other digital testing techniques, such as transition counting, have been ineffective in consistently and accurately locating defects. This is partially caused by the inability to test a circuit in a manner which detects timing related errors.

What would be most desirable is to apply all possible data to all possible inputs as a truth table type signal set while the circuit runs at its full operating speed. The set need not be organized in an orderly fashion as one usually constructs a truth table on paper; the objective is to apply as many of the possible inputs as is feasible with order of application being unimportant. For all but the simplest digital circuits, checking the outputs and intermediate circuit states for such an input set would be impossible by conventional means. With signature analysis, however, a multitude of intermediate and output conditions can be quickly and accurately analyzed. There are two keys to this ability; 1) high speed circuit exercising and 2) data comparison.

When using signature analysis, many input combinations are applied to the unit under test (or are generated by the unit under test) at the circuit's full operating speed, up to 10 MHz. Then responses, at one node, to all of these inputs are compressed into a four-character hexadecimal signature. The four-character hexadecimal display has a total of 12^{16} (16^4) possible states, which in conjunction with the 16 bit input shift register (which also has 2^{16} possible states) and associated feedback provides a unique "compressed" representation of a data stream.

For testing prototype digital devices and for field service of digital devices, the HP 5004 portable Signature Analyzer is a valuable tool. For production testing of digital printed circuit boards, the signature analysis option to the 3060A Board Test System offers simple, quick, and accurate testing of complex digital printed circuit boards.

II. Operation and applications of signature analysis

One of the main advantages of signature analysis over other digital testing techniques is its ability to detect timing related errors. In order to have this ability, the data stream to be checked must be input to the analyzer in a synchronous manner and since we wish to perform this test at the circuit's full operating speed, we make a connection from the system clock to the signature analyzer. A clock phase or control signal for which the testpoint data is stable is chosen for this purpose. The choice of the clock connection point is simplified by the ability to select whether data is accepted on the rising or falling edge of the clock signal. For asynchronous devices, the signature analyzer's clock input will be connected to the clock of the circuit which generates the input signal set (more on this under "Testing ROMs and Combinational Circuits").

While accurate signatures can be obtained by exercising a circuit with a single finite length data stream, it is more desirable to let the circuit be exercised in a continuous loop. We may then read a signature for any one of these complete cycles. In order to perform this function, it is necessary to be able to tell the analyzer what data of the infinite cyclic stream to read for the determination of each signature. This is accomplished by start and stop connections made between the unit under test and the signature analyzer. These two connections set the "time window" during which the analyzer reads data. The choice of these connection points is simplified by allowing the start and/or stop to occur after a high to low or low to high transition of these signals.

The last connection between the unit under test and the analyzer is the analyzer's data input which is programmed to different circuit nodes to check their characteristic signatures. The determination of whether a signature is good or bad is then made by comparing the signatures of the board being tested to the signatures of a known good board (with these known good signatures being stored within the system as demonstrated in Appendix E). Bad signatures may then be traced to the source of a fault by programming the analyzer's data input to various circuit nodes.

The analyzer's data input is fed to a 16 bit shift register with feedbacks. It is very similar to a register used in some pseudorandom code generators. The reason for this stems from information theory which says that the shift register contains a maximum amount of information when all of its possible states occur with equal probability. When this condition is met, the shift register takes on one of 2^{16} possible values with a probability of being in this state of $1/2^{16}$. For input streams ranging from about 16 bits to beyond 2^{16} bits this condition is approached and measured signatures can be compared to correct signatures with the excellent assurance that if the data is correct the measured signature will match the correct signature.* For the reader not previously familiar with the operation of a signature analyzer an illustrative example of its operation is given in Appendix A.

The features of signature analysis provide a flexibility which allows it to be easily applied to many digital devices from combinational circuits to individual ROMs to microprocessor-based boards. As an integral part of the 3060A Board Test System, signature analysis offers simple, accurate and flexible production testing of digital printed circuit boards.

*For a further look at signature analysis accuracy, see "Electronics Magazine", March 3, 1977; Vol. 50; No. 5; p 91.

III. General SA testing techniques

Signature Analysis testing relys on the principle of "exercising" or wiggling circuit nodes - that is forcing a node to change states. The manner by which this is accomplished is relatively unimportant. For example, a counter can generate a truth table type input signal set which is a good exercise for combinational circuits and ROMs. Or if a part of the circuit is designed as a self-test for the final product, this makes a good exercising routine. For exercising microprocessor-based boards, the address bus can easily be made to continuously cycle through its entire address field, exercising a good portion of the board's circuitry. In addition, simple test programs written into ROM may be used to more thoroughly test MPU-based boards.

Once a means of exercising the board is implemented, signatures can be taken at circuit nodes and a bad signature can be traced to its source analogous to the way one traces a bad waveform to its source in an analog circuit. This analogy also applies in relation to testing circuits which employ feedback in that the feedback should be disconnected if one wishes to fully isolate the source of an error. While not all digital circuits contain feedback, sequential circuits, as well as MPU-based boards (with their data busses and interrupts), do have feedback to be disconnected for the purpose of diagnosing faults to the component level.

As with analog signal tracing, several fault isolation techniques exist for digital signature tracing. Expanding the kernel is one technique which is analogous to tracing a test signal from its point of application toward the output. As the testpoint is moved away from the input, there will be a point where the signatures will change from good to bad. The faulty component is thus determined to be one of the components connected to the point where the first bad signature is encountered.

Half-splitting is another technique for fault isolation in which the board test programmer chooses a point in the circuit where failure ahead of or behind this test point is approximately equally likely. In this way, the bad half is determined and then split in half again and again until the fault is isolated.

Expanding the kernel is especially useful for field testing since starting at the input end of a circuit where the probability of component failure is low provides the user with high assurance that the test equipment is set up properly. Half-splitting, on the other hand, is a more efficient technique and therefore recommended for board test programming.

One important difference between analog signal tracing and digital signature tracing should be noted. When tracing waveforms in an analog circuit, one gets clues as to the cause of a fault from waveshape, such as a clipped sinusoid or 60 cycle hum superimposed on the test waveform. When tracing signatures, however, subtle differences between signatures are not useful in the same way that differences in waveshape are useful for analog analysis. If one bit of a long data stream is in error, it is very likely that all four characters of the resulting signature will be different from the characters of the correct signature (this can be seen by examining the examples given in Appendices A & B). The particular characters of a bad signature are, therefore, of little use except for conveying to the tester whether they agree with the correct signature or not and thus whether the signal at the node under test is correct or in error. The SA data input may then be programmed to different circuit nodes with each measured signature telling the user where the fault is in relation to the node being tested.

There are a few special cases where the signature tells the user more than whether the signal at the node under test is good or bad. One case is 0 0 0 0 signature which either indicates a stuck logical 0 or a signal that is low prior to transition of each clock pulse. Another case occurs when a bad signature for one node matches a bad signature for a different node on the board. In this case, there is a good chance that these two points are shorted together, although this condition might be diagnosed by the 3060A's shorts test.

One last signature which conveys information other than whether the signal at the test node is good or not is the V_{cc} signature. When the analyzer's data input is connected to V_{cc}, a continuous string of ones are input to the analyzer's shift register and the resulting signature will take on different values dependent upon the length of the time window (see Appendix B for an illustration of this). Since improper setup of the SA inputs and triggering edges would most likely cause the time window to change, this signature gives information related to the correctness of the start, stop, and clock connection points and triggering edges.

In addition SA tests can be implemented such that the time window is one length for proper circuit operation and a different length (even one clock cycle longer is a sufficient difference) for improper circuit operation. A V_{cc} signature would then differentiate between a good and a faulty circuit.

IV. Testing ROMs and combinational circuits with SA

Combinational circuits and individual ROMs (i.e., ROMs which at this stage of production have not been interfaced with other circuitry) contain no feedback paths.* Because there is no feedback, these circuits can be fault isolated to the component level without the need to design any special signature analysis test capabilities into each printed circuit board. To exercise these circuits, there are several options. If there is circuitry on the board which was designed to be a self-test for the final product, this may be used as an exercise for SA production testing.

There is another exercising possibility which exists if the board to be tested interfaces with another board in the final product which contains a self-test routine. In this case, the board containing the test routine can be mounted on the 3060A's test fixture and connected such that it can exercise each unit under test.

Even if these self-test routines are only intended to provide a go/no go indication for the final product when they are used in conjunction with SA techniques, faults can be isolated to the component level.

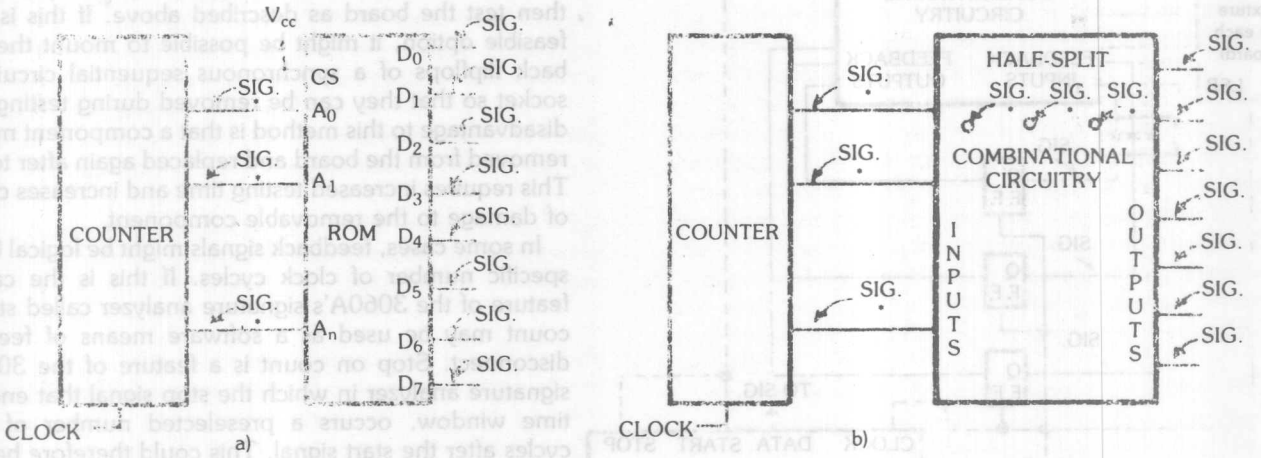
If self-test capabilities were not designed into the product, a counter built into the 3060A test fixture clocked from one of the 3060A's clocks can generate a signal suitable to exhaustively exercise these devices. The number of counter bits required will be equal to the number of inputs for combinational circuits and the number of address bits for ROMs.

The counter input can be applied to a ROM's address inputs and signatures can be taken at the ROM's data output as shown in Figure 1a. If one of these signatures is in error, it is a good practice to have the test program then take signatures on the address inputs of the ROM to determine whether the counter is working and if the ROM inputs are good.

All of these signatures can be taken with the analyzer's start and stop inputs connected to the MSB of the address input to the ROM.

In a similar manner, a counter built into a 3060A fixture can be applied to the inputs of a combinational circuit as shown in Figure 1b. The clock signal may be generated by the 3060A and the analyzer's start and stop connections are made to the MSB of the counter. Signatures may then be taken at the circuit outputs. If one of the output signatures is in error, the board test program can be written to isolate the fault by half-splitting.

*Combinational circuitry which has feedback is classified as sequential circuitry since its outputs are dependent on past inputs.



*If any output signatures are in error, check these points to isolate a fault to the component level.

Figure 1 - Connection of a counter to a) a ROM and b) a combinational circuit, to provide an exercise for signature analysis

V. Testing sequential circuits with SA

Designing with SA in Mind

Sequential circuits are devices for which the output is dependent on past inputs, as well as present inputs. Implementations usually employ feedback which should be disconnected for SA testing if more information is desired than a go/no go indication. If feedback is not disconnected, faults could propagate from their source through the circuit, and then through feedback they could be introduced at the inputs. When this occurs, fault isolation is no longer possible since signatures ahead of, as well as behind, the faulty component would be in error.

When a board is in the early design stages, provision should be made for the disconnection of feedback during testing. This may be implemented with switches, jumpers, or a tristate buffer inserted into the feedback path(s) as shown in Figure 2. Switches and jumpers have the advantage of being positive disconnecions and the disadvantage of requiring extra circuit handling. A tristate buffer could be employed such that an input driven from a 3060A driver can put the buffer into its high impedance state. The disadvantage of this scheme is that if the buffer should fail, signals could be present at the feedback inputs. Thus, if this method of feedback disconnection is employed, it is recommended that an early part of the test program check the buffer to see that none of its outputs are stuck high or low (this could be done with the 3060A's static test capabilities). Once feedback paths have been disabled, a sequential circuit is basically a combinational circuit and can be tested as such.

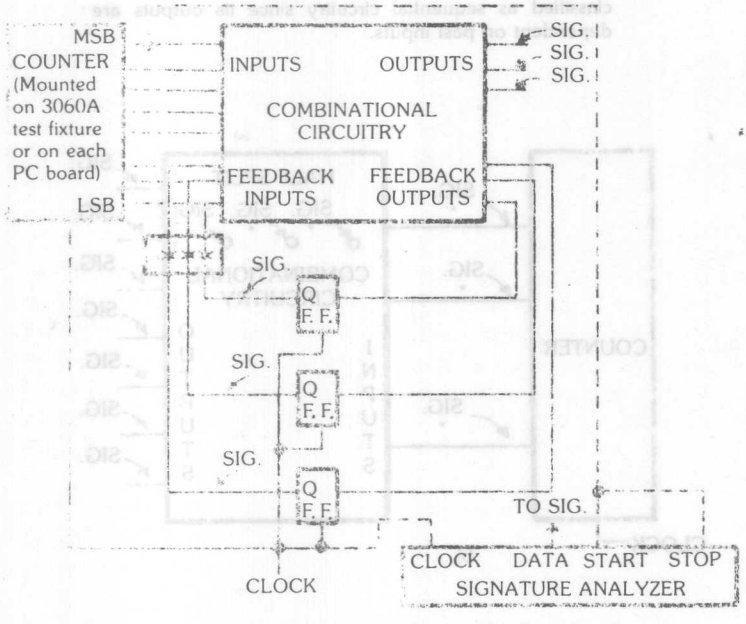


Figure 2 - Method for testing synchronous sequential circuits. Feedback disconnections are shown by X. Connections which are made for the purposes of SA testing are shown with - - - - lines.

If a synchronous sequential circuit is being tested, the circuit's own clock can be used to clock a counter built into the 3060A test fixture which can then be connected to the circuit inputs (including the now disconnected feedback inputs).

If an asynchronous sequential circuit is being tested, the counter can be built into the test fixture and clocked from a 3060A clock. In both cases, the signature analyzer's start and stop inputs may be programmed to the most significant bit of the counter and signatures can be taken at the circuit outputs. If a bad signature is found, half-splitting may be used to isolate the fault.

Retrofitting

When given a sequential circuit which was designed without signature analysis testing in mind there are several testing options. The most desirable option is to add a method for disconnecting any feedback path(s) and then test the board as described above. If this is not a feasible option, it might be possible to mount the feedback flipflops of a synchronous sequential circuit in a socket so that they can be removed during testing. One disadvantage to this method is that a component must be removed from the board and replaced again after testing. This requires increased testing time and increases chance of damage to the removable component.

In some cases, feedback signals might be logical 0 for a specific number of clock cycles. If this is the case, a feature of the 3060A's signature analyzer called stop on count may be used as a software means of feedback disconnect. Stop on count is a feature of the 3060A's signature analyzer in which the stop signal that ends the time window, occurs a preselected number of clock cycles after the start signal. This could therefore be used to end the time window just before any signals are fed back.

If none of the above choices is possible, signatures can still be taken to determine whether the board is good or faulty, but fault isolation capabilities have been sacrificed.

VI. Testing microprocessor-based boards with SA

Disconnecting feedback paths

When designing a microprocessor-based board which will be tested with signature analysis, it is desirable to design a method of feedback disconnect into the circuit to aid in fault isolation. There are ways in which a MPU-based board can be tested (and even partially fault isolated) without feedback disconnect provisions; but since there are other drawbacks to these methods, consideration of them is deferred to a discussion of retrofitting.

Interrupt lines and the data bus input to a microprocessor are feedback paths for which one of several disconnect methods should be chosen. These methods vary in terms of board space required, reliability of operation, and simplicity of use. Choosing the most suitable method is, therefore, dependent on the particular application and its associated restrictions. Generally, most SA test capabilities can be built into the 3060A's fixture (counters, a PROM with SA test programs, etc.). As more testing related implementation is put onto the test fixture (as opposed to being employed on each individual printed circuit board), ease of field testing is reduced.

A DIP jumper or switches are the most positive means of disconnecting the data bus input to the processor. An alternative which can be used if a tristate buffer has been added to the data bus for other reasons (such as a heavy bus load) is to use this buffer to disable the data bus feedback path. By adding an and gate to the buffer's disable input, the buffer can be driven to its high impedance state with a 3060A driver. Then, when being field tested, the buffer may be disabled with a jumper wire or with a switch.

If there is no buffer for other reasons one could be added specifically for the purpose of feedback disablement. In either case, there is a chance that the buffer output could be stuck or not obey the disable instruction. It is, therefore, recommended that the 3060A test program check the data bus to see that the buffer outputs are not stuck high or low.

Interrupt lines are most easily disconnected by an interrupt masking instruction. Or, if desired, they could be disconnected directly with a jumper or switch connected to the processor's interrupt input.

Exercising the board

Once these feedback paths have been disconnected, we want to consider how to exercise the circuit. The simplest way to do this is to force the MPU to free-run.

In relation to microprocessors, this involves getting the processor to continuously cycle through its entire address field.* This can be accomplished by applying an instruction to the processor's now disconnected data input which causes the processor's program counter to increment. The processor will then check its data bus where it will again see the "free-run" instruction and again the program counter will increment, continuing the cycle. In this way the processor's address bus scans its entire address field, thus applying a truth table type signal set to the address bus.

The choice of what instruction will be applied to the processor to get it to free-run is not critical - basically any instruction which causes the program counter to increment and the processor to thus look for the next instruction will suffice. Some instructions, however, are easier to implement than others. The application of a free-run instruction can be accomplished directly with drivers on the 3060A so that no hardware needs to be added to the circuit. For increased field testability, however, a few parts may be added which can be used to force the free-run condition in the field. As shown in Figure 4 "pull up" resistors are permanently connected from each of the data bus lines to the V_{cc} supply. Then diodes are connected from the appropriate data lines through a "free-run switch" to ground. These connections are made on the side of the data bus disconnect which is toward the processor. Then when the data bus is opened and the switch grounding the diodes is closed, the free-run instruction is applied to the MPU's data input. Implementations of this for several different microprocessors are given in Appendix C.

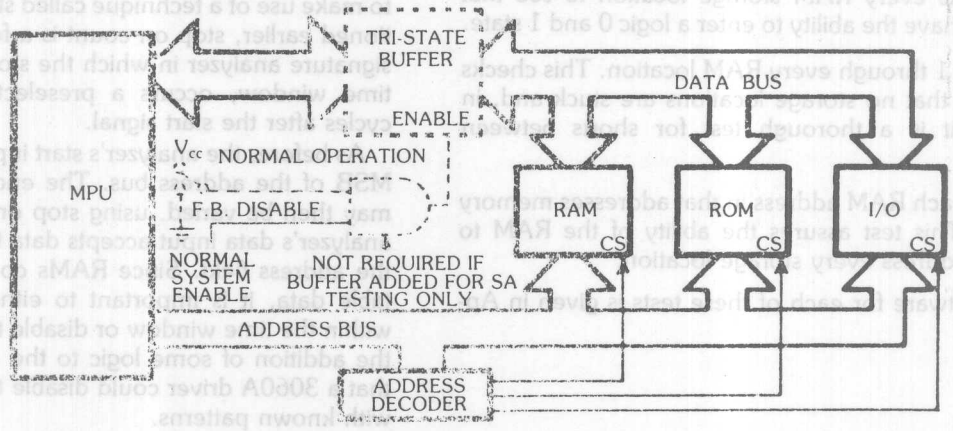


Figure 3 - Generalized microprocessor based circuit which employs a buffer in the data bus as a means of feedback disablement.

*For a more general definition of free-running see HP Application Note 222, p6.

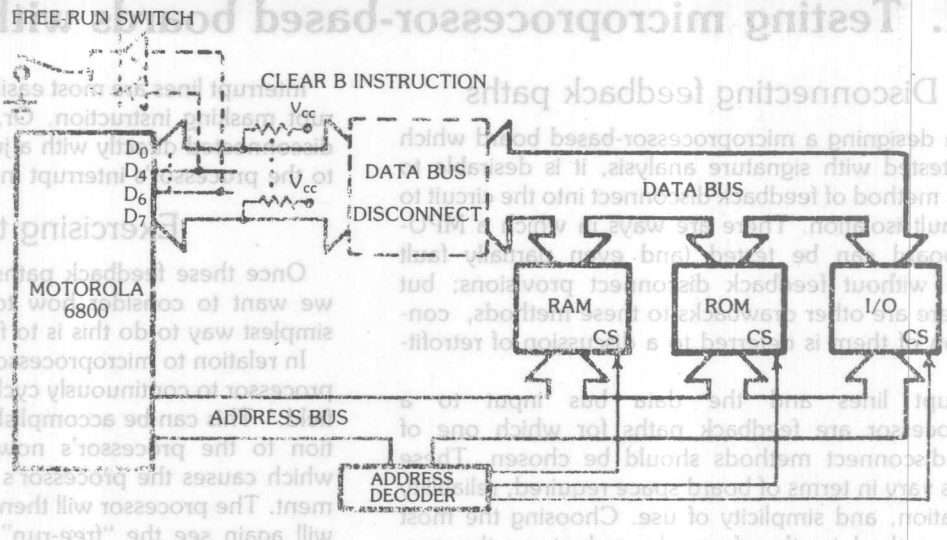


Figure 4 - Method for free-running a microprocessor which can be used for both field and production testing.

Output ports are exercised by free-running. Signatures can therefore be taken at these ports and on their peripheral circuitry. Input ports, however, are not exercised by free-running. One method for exercising these ports is to tie outputs back to inputs with 3060A relays.

The major parts yet untested are the RAMs. These may be tested by now reconnecting the data bus to the processor. Then a simple test program written into a small portion of ROM (or in a PROM on the test fixture) can write data into RAM and read it out again. Checking the data bus signatures for the read cycle of this exercise verifies the operation of the RAM.

There are many RAM test programs designed to verify different aspects of a RAM's performance. Here are a few examples.

1. Store the contents of ROM into RAM. This provides a somewhat random-check on the addressing ability of the RAM and the operation of its storage locations.

2. Store a checkerboard pattern of 1's and 0's into RAM and then store in the complement of this pattern. This checks every RAM storage location to see that they each have the ability to enter a logic 0 and 1 state.

3. Walk a 1 through every RAM location. This checks to be sure that no storage locations are stuck and, in addition, it is a thorough test for shorts between locations.

4. Store each RAM address in that addresses memory location. This test assures the ability of the RAM to properly address every storage location.

Example software for each of these tests is given in Appendix D.

Once the processor is free-running, the board's operation can be simply checked with the techniques of half-splitting or expanding the kernel. Expanding the kernel will be used here for illustrative purposes although half-splitting might be preferred to increase testing efficiency. The processor's addressing field may be checked by programming the signature analyzer's start and stop inputs to the address bus's MSB while checking signatures on all of the address lines. With these same start, stop, and clock connections, signatures may be taken at the address decoder outputs to check its operation.

The start and stop connections may then be changed to shorten the time window so that the analyzer only inputs data to its shift register during a portion of the address field corresponding to a single device. This may be implemented by connecting the start and stop inputs to the chip select of the individual device to be tested. If this is done and the analyzer's data input is programmed to the data bus (on the side of the feedback, disconnect away from the processor), the contents of each ROM is verified.

Another method for "windowing in" on a bad ROM is to make use of a technique called stop on count. As mentioned earlier, stop on count is a feature of the 3060A's signature analyzer in which the stop signal that ends the time window, occurs a preselected number of clock cycles after the start signal.

As before, the analyzer's start input is connected to the MSB of the address bus. The end of the time window may then be varied, using stop on count, such that the analyzer's data input accepts data for various portions of the address field. Since RAMs could contain unpredictable data, it is important to either not include RAMs within the time window or disable the RAMs (possibly by the addition of some logic to the address decoder such that a 3060A driver could disable them) or fill the RAMs with known patterns.

Retrofitting boards for SA

When retrofitting a microprocessor-based board for signature analysis testability there are several considerations to be made. Methods of feedback disablement and circuit exercising should be chosen.

Devices which were not designed with signature analysis in mind are often designed modularly such that the microprocessor and a few individual components are located on a separate board from ROM and RAM. In this case, the data bus input to the processor has already been disconnected. The processor board can then be tested by free-running the processor (this can be forced from drivers on the 3060A) while taking the signatures of the address bus, and other devices on this board.

RAMs located on boards separate from the processor can be tested by mounting the processor board on the test fixture. A PROM containing the SA test program could also be included on the test fixture. RAM boards are thus tested under the control of the microprocessor with which it will interface in the final product.

ROM boards could also be tested under processor control by mounting a processor board on the test fixture, or they could be tested as described in the section on testing ROMs by mounting a counter on the test fixture.

If the board to be retrofit was not modularly designed there are several other options.

To disconnect the data bus input to the processor one could add either a DIP jumper, switches, or a tristate buffer to the bus. The tristate buffer has the advantage that it can be driven to its high impedance state from a 3060A driver. This driver can be controlled by the test program, thus saving production testing time over other disconnect means.

If there is not board space to add one of these disconnect methods, a few single wire jumpers can be added to the address decoder such that when these jumpers are pulled, all bus-connected devices are put into their high impedance state. This method limits free-run testability to the processor's address counter alone, since the address decoder and ROMs are now disabled.

Once the processor's operation is verified by this method, other components may be tested (and to a certain extent fault isolated) by software signature analysis routines written into memory which are executed after reconnecting the address decoder jumpers. This is one instance where a fault may be partially isolated without the processor's data bus input being disabled. If a checksum is performed on each ROM in such a way that an unused address bit toggles at the start and again when an incorrect checksum is found, a variable length time window may be created for the signature analyzer. If the analyzer's data input is then programmed to V_{cc}, the signature will be different depending on the length of the time window and thus depending on which ROM has failed. (See Table 1)

For the greatest field testability, SA test programs should be written into a ROM which is already on the board. Often the space at the end of a ROM already on the board is enough for a thorough test program. If this is not the case, a separate ROM might be added to the board.

If there is not enough space for this, there are several other options. For example, a PROM (which contains an SA test routine) could be included on the 3060A fixture.

Another option which could be cost effective for some users is to store an SA test stimulus program in a logic pattern generator, such as the HP 8170A.

The 8170A is a programmable word generator in which an SA test routine could be stored and then executed by connecting its outputs to scanner relays on the 3060A. Once the program has been entered into the 8170A it can be used to directly drive a disabled data bus. In this way a microprocessor-based board may be exercised at speeds up to 2 MHz, providing the 8170A is located close enough to the scanner such that cable capacity is not excessive.

Table 1

Motorola 6800 Code for performing a checksum on two ROMs ("ROM1" and "ROM2"). Address line A15 is toggled at the start. A variable length time window is created by again toggling A15 when an incorrect checksum is found, or when the test is completed. The V_{cc} signature will take on one of only three possible values depending on whether 1) ROM1 is bad, 2) ROM2 is bad, or 3) both ROMs are good.

MEM. ADDRESS LOCATION	LABEL	MNEMONIC	LOCATION OR #	EXPLANATION
3800	CSR1	FCB	$D2	Check sum of ROM1
3801	CSR2	FCB	$8C	Check sum of ROM2
	STSP	EQU	$8000	Label used to toggle A15 for SA start and stop
	BESA	STA A	STSP	
		LDX	#$3801	Load index register with the starting address of ROM1
		CLR A		Clear accumulator for checksum
	ROM1	ADD A	0,X	Add contents of ROM to Accumulator A.
		INX		Increment index register to check next location
		CPX	#$4000	Compare the index register with the end of ROM1.
		BNE	ROM1	Branch to "ROM1" if all ROM1 locations have not been checked
		STA A	$00	Output checksum onto data bus
		CMP A	CSR1	Compare accumulator A with ROM1 checksum
		BNE	BESA	Branch to "BESA" to end time window if not equal
	ROM2	LDX	#$4000	Same as above except ROM1's are changed to ROM2's
		CLR A		
		ADD A	0,X	
		INX		
		CPX	#$4200	
		BNE	ROM2	
		STA A	$00	
		CMP A	CSR2	
		BNE	BESA	
		NOP		Extend time window one step
		BRA	BESA	If both checksums are correct then end time window

VII. Summary

The versatility of signature analysis allows it to quickly and accurately isolate digital faults to the component level. As an option to the 3060A Board Test System, SA is particularly flexible in that it is especially easy to implement. SA testing is further simplified with the aid of 3060A capabilities and features such as programmable drivers, stop on count, and the ability to mount components or PC boards on the test fixture.

An average of incremental design and development costs for designing SA testability into five HP products was about one percent while incremental material costs were less than one percent. The additional time required to program signature analysis fault isolation tests on the 3060A is also a very small fraction of the total product development.

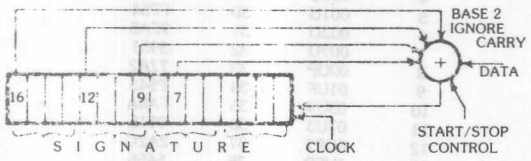
When one considers the fault isolation capabilities and extremely consistent characterization of good and faulty devices, these costs are recovered many times over for normal use of the 3060A.

VIII Appendices

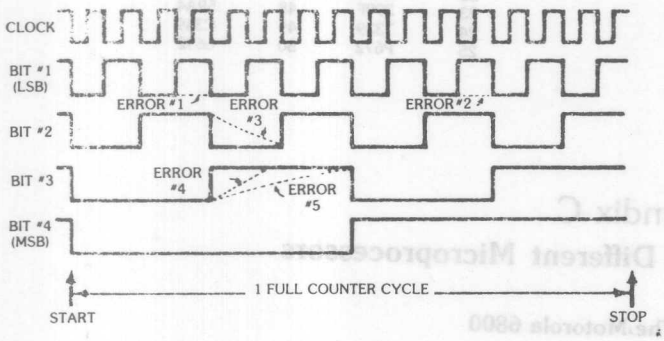
Appendix A

How the Analyzers Shift Register Works

Shown below is a model of a signature analyzer.



For illustrative purposes let us take a four-bit counter and look at some signatures associated with its four outputs. Then, let us see what these signatures would be if possible errors (shown ----lines) are introduced.



If the analyzer's clock input is connected to the clock signal shown, and if the analyzer's start and stop connections are made to the MSB of the counter, the time window (i.e., the portion of the data stream used for determination of each individual signature) will consist of one full counter cycle. This is true providing that the following triggering edges are selected: ⌟ for clock, and ⌝ for start and stop.

Several signatures we might check are:

a) Bit 1 with no errors
b) Bit 1 with error #1 introduced
c) Bit 1 with error #2 introduced
d) Bit 2 with no errors
e) Bit 2 with error #3 introduced
f) Bit 3 with no errors
g) Bit 3 with error #4 introduced
h) Bit 3 with error #5 introduced

For instructive purposes we will derive the signatures for signals a and b. This derivation is shown below with data input "a" shown first while the changes error #1 introduces (to produce signal b) are shown shaded.

CLOCK CYCLES AFTER "START"	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	DATA INPUT
1																0	0
2															0	1	1
3														0	1	0	0
4													0	1	0	1*	1*
5												0	1	0	1*	0	0
6											0	1	0	1*	0	1	1
7										0	1	0	1*	0	1	0	0
8									0	1	0	1*	0	1	0	1	1
9								0	1	0	1*	0	1	0	1	1	0
10							0	1	0	1*	0	1	0	1	1	1	1
11						0	1	0	1*	0	1	0	1	1	1	0*	0
12					0	1	0	1*	0	1	0	1	1	1	0*	1	1
13				0	1	0	1*	0	1	0	1	1	1	0*	1	0*	0
14			0	1	0	1*	0	1	0	1	1	1	0*	1	0*	0	1
15		0	1	0	1*	0	1	0	1	1	1	0*	1	0*	0	0	0
16	0	1	0	1*	0	1	0	1	1	1	0*	1	0*	0	0	1*	1
RESULTING SIGNATURE:	5F				5			H*			1*						

In a similar manner, signatures for input signals a through h can be determined. Shown below are these input data streams and the resulting signatures. Changes caused by the errors shown in the counter timing diagram are again shown shaded.

CLOCK CYCLES AFTER "START"	SIGNAL PRESENTED TO DATA INPUT OF ANALYZER							
	a Bit 1 No Error	b Bit 1 Error #1	c Bit 1 Error #2	d Bit 2 No Error	e Bit 2 Error #3	f Bit 3 No Error	g Bit 3 Error #4	h Bit 3 Error #5
1	0	0	0	0	0	0	0	0
2	1	1	1	0	0	0	0	0
3	0	0	0	1	1	0	0	0
4	1	1*	1	1	1	0	0	0
5	0	0	0	0	0*	1	1*	1*
6	1	1	1	0	0	1	1	1*
7	0	0	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	0
11	0	0	0	1	1	0	0	0
12	1	1	1*	1	1	0	0	0
13	0	0	0	0	0	1	1	1
14	1	1	1	0	0	1	1	1
15	0	0	0	1	1	1	1	1
16	1	1	1	1	1	1	1	1
RESULTING SIGNATURE	55H1	45U8	55F1	334U	3C5C	0U16	0702	0308

Note that minor errors in data cause major differences in the resulting signatures. Note also (from a, b, and c) that the same type of error occurring at different times results in very different signatures. In addition, from signals f, g, and h it is apparent that the same error occurring with greater severity (h) causes a different signature from the signature which results with the same type of error of lesser severity (g), and that they both are different from the correct signature (f).

Appendix B

Shifting a Continuous String of Ones into the Analyzer

Clocking a continuous string of 0's into the analyzer's shift register will always result in a 0 0 0 0 signature. Clocking a continuous string of 1's into the analyzer (by connecting the data probe to a V_{cc}), however, does not always result in the same signature. Feedback from within the analyzer's shift register (as shown in Appendix A) assures this "randomizing" of non-zero input data so that a maximum amount of information is conveyed in a signature. Because of this fact, connecting the analyzer's data input to logic 1 will check the time window and, thus, the start, stop, and clock connection points and triggering edges. If one of these is connected incorrectly, the signature will most likely be different than the signature which would result if the proper connections were made.

Shown below are the signatures which result from shifting one through fifty 1's into the analyzer. Note that all 50 signatures are different and that one extra or one fewer clock pulses than intended causes a signature different from the desired signature. In fact, any one signature will not repeat until 65,536 clock cycles later. Note also that in this case, the pattern present at the analyzer's data input is far from random (i.e., constant logic 1) yet soon after 16 clock cycles, the signatures appear random. In practice, much greater than 16 bits will be shifted into the analyzer such that the state which the shift register is in when the "stop" signal occurs will be occupied with a probability of approximately $1/2^{16}$.

# of 1's Clocked In	Resulting Signature	# of 1's Clocked In	Resulting Signature
1	0001	26	FFP5
2	0003	27	99FA
3	0007	28	3395
4	000U	29	672A
5	001U	30	FP54
6	003U	31	9FA8
7	007U	32	3951
8	00UP	33	72A2
9	01UF	34	P545
10	03U9	35	FA8A
11	07U3	36	9515
12	0UP7	37	2A2C
13	1UFP	38	5456
14	3U9F	39	A8AF
15	7U39	40	5159
16	UP73	41	A2C3
17	UFP6	42	4566
18	U9FF	43	8AFH
19	U399	44	159A
20	P733	45	2C34
21	FP67	46	5669
22	9FFP	47	AFH2
23	399F	48	59A4
24	7339	49	C349
25	P672	50	6692

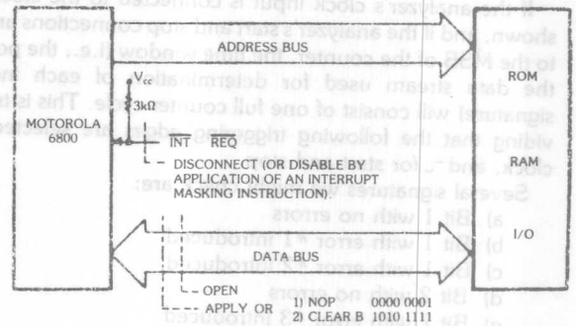
Appendix C

How to Free Run Several Different Microprocessors

Free-Running The Motorola 6800

The free-run instruction may be applied by using pull-up resistors and diodes (as shown in Figure 4), or by use of 3060A drivers.

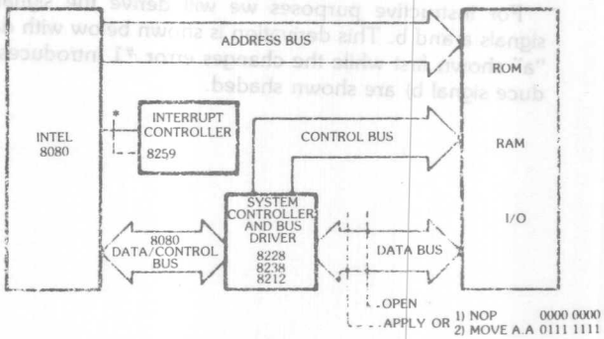
START		STOP		CLOCK		TAKE SIGNATURES ON:
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	
A15	⌝	A15	⌝	0_2	⌟	Address Bus
A15	⌝	A15	⌝	0_2	⌟	Any wiggled node which is stable during ⌟ of 0_2.
ROM C.E.	⌟	ROM C.E.	⌝	0_2	⌟	Data Bus (To isolate individual ROM)



Free-Running The Intel 8080

The free-run instruction may be applied by using pull-up resistors and diodes or by use of 3060A drivers.

START		STOP		CLOCK		TAKE SIGNATURES ON:
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	
A15	⌝	A15	⌝	DEBIN	⌝	Address Bus
A15	⌝	A15	⌝	STATUS STROBE	⌝	8080 Control Lines
A15	⌝	A15	⌝	DEBIN	⌝	Any wiggled node which is stable during ⌟ of DEBIN.
ROM C.S.	⌟	ROM C.S.	⌝	DEBIN	⌝	Data Bus (To isolate individual ROM)
A15	⌝	A15	⌝	DEBIN	⌝	8028 Control Bus

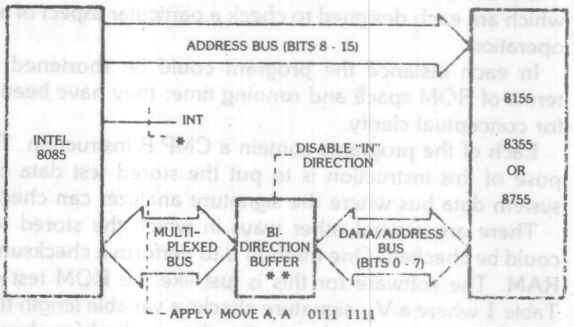


*Disconnect (or disable by application of an interrupt masking instruction).

Free-Running The Intel 8085 With Special Memory I/O Chips

The free-run instruction should be applied with the use of 10k pull-up resistors except for a fast diode pull-down from AD_7 to RD.

START		STOP		CLOCK		TAKE SIGNATURES ON:
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	
A15	↓	A15	↓	ALE	↑	Address Bus
A15	↓	A15	↓	ALE	↑	Any wiggled node which is stable during ↑ of ALE
ROM C.E.	↑	ROM C.E.	↓	RD	↑	Data/Address Bus



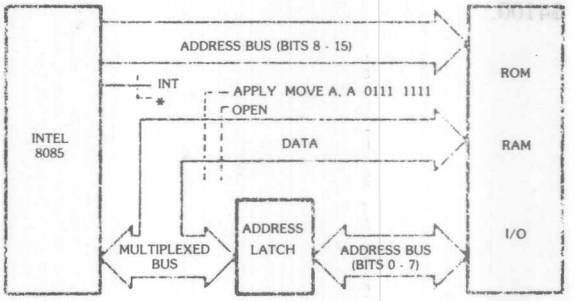
***Disconnect (or disable by application of an interrupt masking instruction).**

****T.I. 74LS245, or T.I. 74L5 243, or National 81LS95; or plug-in jumper for normal operation and "out" only buffer for SA testing.**

Free-Running The Intel 8085 With Conventional Memory And I/O Chips

The free-run instruction should be applied with the use of 10k pull-up resistors except for a fast diode pull-down from AD_7 to RD.

START		STOP		CLOCK		TAKE SIGNATURES ON:
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	
A15	↓	A15	↓	RD	↑	Address lines (outboard of latch)
A15	↓	A15	↓	RD	↑	Any wiggled node which is stable during ↑ of RD
A15	↓	A15	↓	ALE	↑	Address lines (inboard of latch)
ROM C.E.	↑	ROM C.E.	↓	RD	↑	Data lines (To isolate individual ROM)

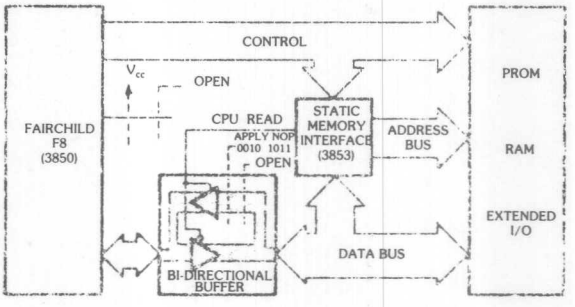


***Disconnect (or disable by application of an interrupt masking instruction).**

Free-Running The Fairchild F8 With External Static Memory

The free-run instruction may be applied by using pull-up resistors and diodes, or by use of 3060A drivers.

START		STOP		CLOCK		TAKE SIGNATURES ON:
CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	CONNECTION POINT	EDGE	
A15	↑	A15	↑	WRITE	↓	Address Bus
A15	↑	A15	↑	WRITE	↓	Any wiggled node which is stable during ↓ of WRITE
A15	↑	Vary STOP on count until fault isolated		WRITE	↓	Data Bus (To isolate individual faults)



Appendix D

Example RAM Test Programs

Shown below are four examples of Motorola 6800 code which are each designed to check a particular aspect of a RAM's operation.

In each instance the program could be shortened both in terms of ROM space and running time; they have been written for conceptual clarity.

Each of the programs contain a CMP B instruction. The purpose of this instruction is to put the stored test data onto the system data bus where the signature analyzer can check it.

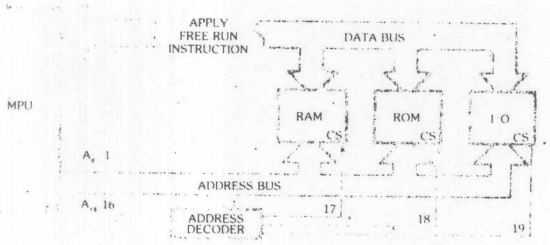
There are several other ways in which the stored test data could be checked. One method is to perform a checksum on the RAM. The software for this is just like the ROM test given in Table 1 where a V_{cc} signature checks a variable length time window for a correct checksum. Another method for checking the test data stored in RAM is to have the processor do a comparison of what is stored against what it recalls from the same location. Again, a variable length time window can provide a fault indication for the signature analyzer.

For each of these examples it is assumed that the RAM to be tested occupies the address locations $0000 to $0100. In addition, test #1 assumes that there is ROM at locations $4000 to $4100.

LABEL	MNEMONIC	LOCATION OR #	EXPLANATION
			STORE ROM IN RAM TEST
	SEI		Set interrupt mask to avoid destruction of stack data used in this test.
	CLX		Clear index register for use as a loop counter and address pointer.
	LDS	$3FFF	Load stack with $4000 - 1 (the first ROM address minus one).
TST 1	PUL A		Stack pointer increments and contents of ROM pointed to by stack are pulled into accumulator A.
	STA A	0,X	Store the contents of accumulator A in the RAM location addressed by the index register.
	CMP B	0,X	Output stored test data to data bus for checking with signature analyzer.
	INX		Increment index register to check next location.
	CMPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 1	Branch to "TST 1" if all RAM locations are not yet checked.
			CHECKERBOARD AND INVERSE CHECKERBOARD TEST
	CLX		Clear index register for use as a loop counter and address pointer.
	LDA A	#$55	Load accumulator A with the binary pattern 0101 0101.
	LDB B	#$AA	Load accumulator B with the binary pattern 1010 1010.
TST 2	STA A	0,X	Store the binary pattern 0101 0101 in the RAM location addressed by the index register.
	STA B	1,X	Store the binary pattern 1010 1010 in the RAM location addressed by the index register plus one.
	CMP B	0,X	Output the stored 0101 0101 pattern to the data bus for checking with signature analyzer.
	CMP B	1,X	Output the stored 1010 1010 pattern to the data bus for checking with signature analyzer.
	INX		Increment the index register.
	INX		Increment again since two patterns were stored.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 2	Branch to "TST 2" if all RAM locations are not yet checked.
	COM B		Complement the 1010 1010 pattern in accumulator B to obtain 0101 0101.
	COM A		Complement the 0101 0101 pattern in accumulator A to obtain 1010 1010.
	BMI	TST 2	If COM instructions have been executed once the N flag will be set and test will continue by branching to "TST 2". The second time through N will be 0 and branch will not occur.
			WALKING 1 TEST
	CLC		Clear carry flag for use.
	CLX		Clear index register for use as a loop counter and address pointer.
TST 3	LDA A	#$01	Load accumulator A with the binary pattern 0000 0001.
AGAIN	STA A	0,X	Store this pattern in the RAM location pointed to by the index register.
	CMP B	0,X	Output stored pattern to the data bus for checking with signature analyzer.
	ASL A		Arithmetic shift left to walk the one across the word.
	BCC	AGAIN	If carry flag is set, the one has walked across the whole word. If not, branch to "AGAIN".
	INX		Increment index register.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 3	Branch to "TST 3" if all RAM locations are not yet checked.
			ADDRESS TEST
	CLX		Clear index register for use as a loop counter and address pointer.
	CLA A		Clear accumulator A for use as a counter.
TST 4	STA A	0,X	Store the contents of accumulator A in the RAM location addressed by the index register.
	CMP B	0,X	Output stored address to the data bus for checking with signature analyzer.
	INX		Increment the index register.
	INC A		Increment accumulator A.
	CPX	#$0100	Compare the index register with the end of the RAM being tested.
	BNE	TST 4	Branch to "TST 4" if all RAM locations are not yet checked.

Appendix E
An Example Board Test Program

Shown below is a microprocessor-based circuit which is assumed to be free-running.



The following is a board test program which stores the correct signatures of nodes 1-19 of a known good board. The second part of the program then compares the signatures of units to be tested with the stored good signatures.

There are three basic BTL commands used in this SA test.

1) saset (insert in these parentheses parameters specifying CLOCK START and STOP inputs and edges) 2) mcon (insert relay number in these parentheses) 3) sig (insert in these parentheses the the parameters including the expected signature)	- Used to set up the inputs for SA - Connects the SA DATA input to the node where the signature is to be taken. - Causes the signature to be taken an input to the calculator.
PART 1 0: dim A$ [19,4]	- define A$, an array in which the 19 four-character signatures will be entered.
1: asgn "SIGS", 1, 0, Z; if Z; open "SIGS", 1	- opens data file "SIGS" on diskette for storage of signatures. "SIGS" is assigned to be file #1.
2: rcv ref 1, .8, 2, 2	- defines high and low reference thresholds at .8V and 2.0V.
3: saset 1,"F",1,"F",1,"F"	- CLOCK input #1, START input #1 and STOP input #1 are selected with falling triggering edges. (SA set-up)
4: for I = 1 to 19	- sets up loop to take 19 signatures.
5: (I + 11)/100 + .0044 →N	- converts node number to multiplex relay number - assumes nodes 1-19 are wired to column 44 row 12-31, respectively, of the test fixture.
6: mcon N	- closes the selected multiplex relay to connect the 3060A SA circuitry to the node under test.
7: sig "GOOD SIGNATURE", "*****", A$[I,1]	- takes the signature and stores it in row I of array A$.
8: next I	- end of loop.
9: sprt 1, A$	- stores (serial prints) the known good signatures on diskette in file 1 (SIGS).
PART 2 0: dim B$[19,4], C$[10]	- dimensions arrays B$ and C$.
1: rcv ref 1, .8, 2, 2	- defines high and low reference thresholds at .8V and 2.0V.
2: asgn "SIGS"	
3: sread 1, B$	- transfers (serial reads) good signature on diskette to the array B$ in the calculator.
4: saset 1,"F",1,"F",1,"F"	- CLOCK input #1, START input #1, and STOP input #1 are selected with falling triggering edges.
5: for I = 1 to 19; fxd 0	- sets up loop to take 19 signatures.
6: "Node #" →C$; str (I) →C$[7]	- stores a label including the node number. This label will be used by statement 8 to designate the number of a failing node.
7: (I →11)/100 + .0044→N	- converts node number to multiplex relay number - assumes same fixture wiring as stated (PART 1, statement 5).
8: mcon N	- closes the selected multiplex relay to connect 3060A SA circuitry to the node under test.
9: sig C$, B$ [I], D	- takes the signature at the node under test and compares it to the correct signature stored in B$. If they are different, the system will print an error message indicating a bad signature was found on "Node #I". The bad signature is also printed out. A number indicating whether the test passed is stored in D which can be checked for troubleshooting decisions in a fault isolation algorithm.
10: next I	- end the loop.

Appendix E

An Example Board Test Program

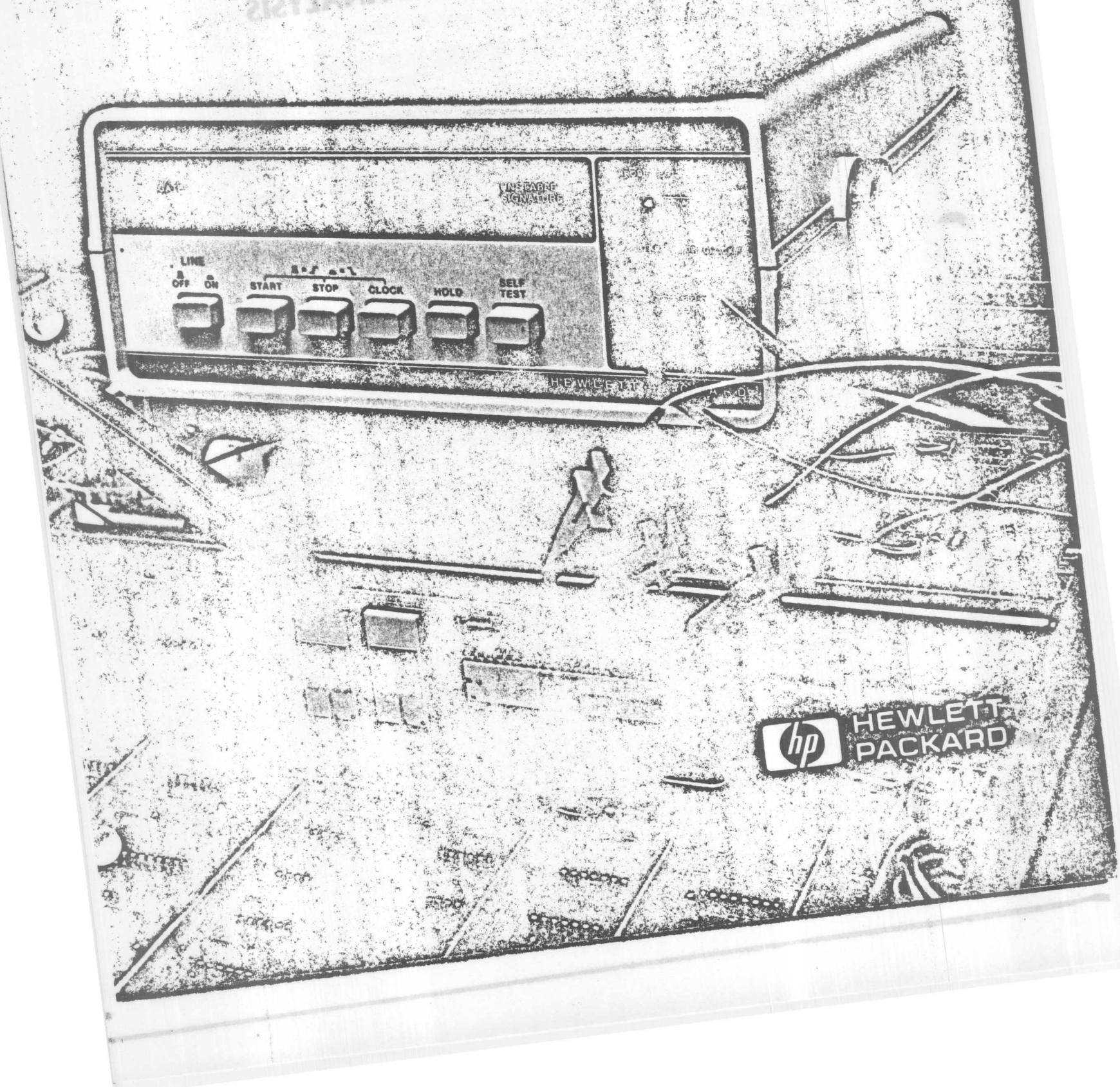
Shown below is a microprocessor-based circuit which is assumed to be free-running.

The following is a board test program which stores the correct signatures of nodes 1-19 of a known good board. The second part of the program then compares the signatures of units to be tested with the stored good signatures.

There are three basic BTL commands used in this SA test:

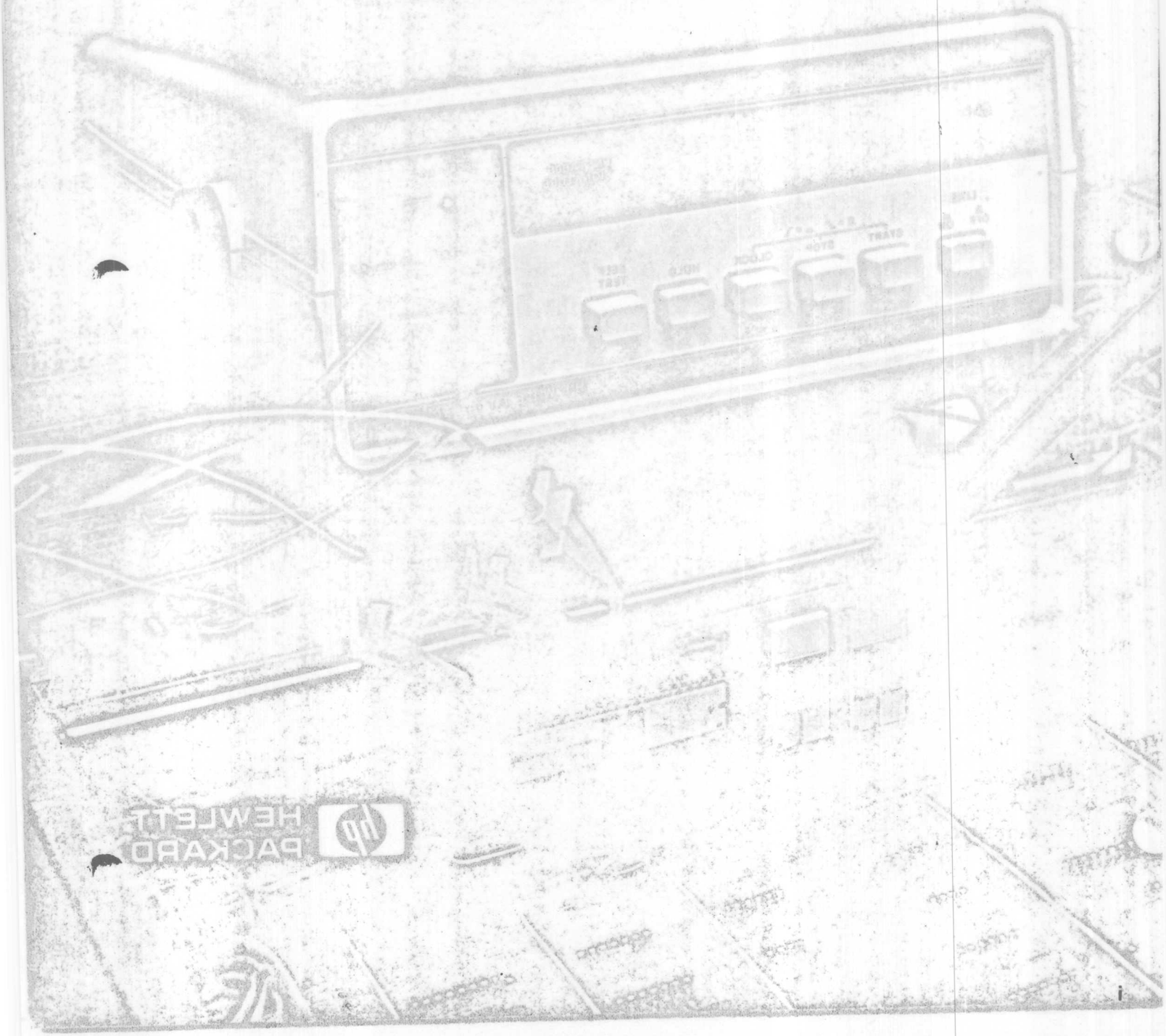
Application Note 222-2

APPLICATION ARTICLES ON SIGNATURE ANALYSIS



APPLICATION NOTE 222-2

APPLICATION ARTICLES ON SIGNATURE ANALYSIS



FORWARD

ABOUT DIGITAL TROUBLESHOOTING

Microprocessors have revolutionized your product line. Your products are smarter, faster, friendlier and more competitive because they take advantage of μP-based control and computation. They are also harder to build, harder to test and harder to fix when they fail. Complex bus structures and timing relationships have practically obsoleted the scope/voltmeter signal tracing techniques so effective on analog products. The need to enhance the testability and serviceability of your digital products is acute. So is the need for specialized digital troubleshooting equipment.

ABOUT SIGNATURE ANALYSIS

To address these needs, Hewlett-Packard has developed the Signature Analysis technique, as well as a Signature Analyzer product line, for component-level troubleshooting of microprocessor-based products. A Signature Analyzer detects and displays the unique digital signatures associated with the data nodes in a circuit under test. By comparing these actual signatures to the correct ones, a troubleshooter can back-trace to a faulty node. By designing or retrofitting S.A. into digital products, a manufacturer can provide manufacturing test and field service procedures for component-level repair, without dependence on expensive board-exchange programs.

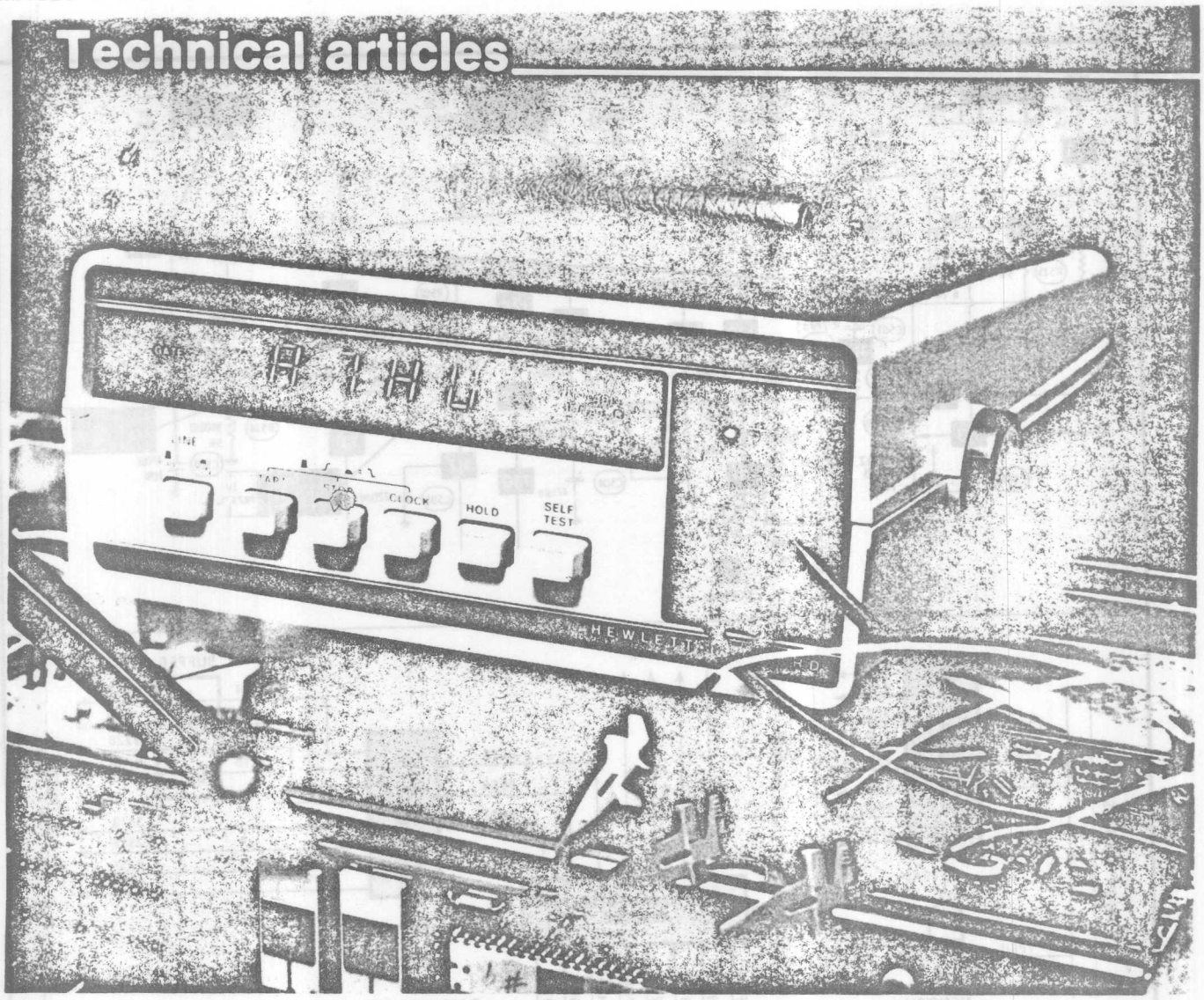
ABOUT THIS PUBLICATION

This is a collection of eight technical articles on Signature Analysis applications. It is intended to assist you in designing or retrofitting your digital products for S.A. troubleshooting. Check the annotated Table of Contents and choose those articles which are of interest in your application.

ABOUT OTHER PUBLICATIONS

Application Note 222-0, HP Publication 02-5952-7593, is a complete index to the latest Signature Analysis publications. It lists all other application notes currently available in the AN 222 series about Signature Analysis. They cover a wide range of interests, from how to design or retrofit Signature Analysis into digital systems, to the cost reductions that can be expected in production test and field service by doing so. It also lists all data sheets for the complete line of Hewlett-Packard Signature Analysis products, plus other related publications about digital troubleshooting.

TABLE OF CONTENTS



Hexadecimal signatures identify troublespots in microprocessor systems

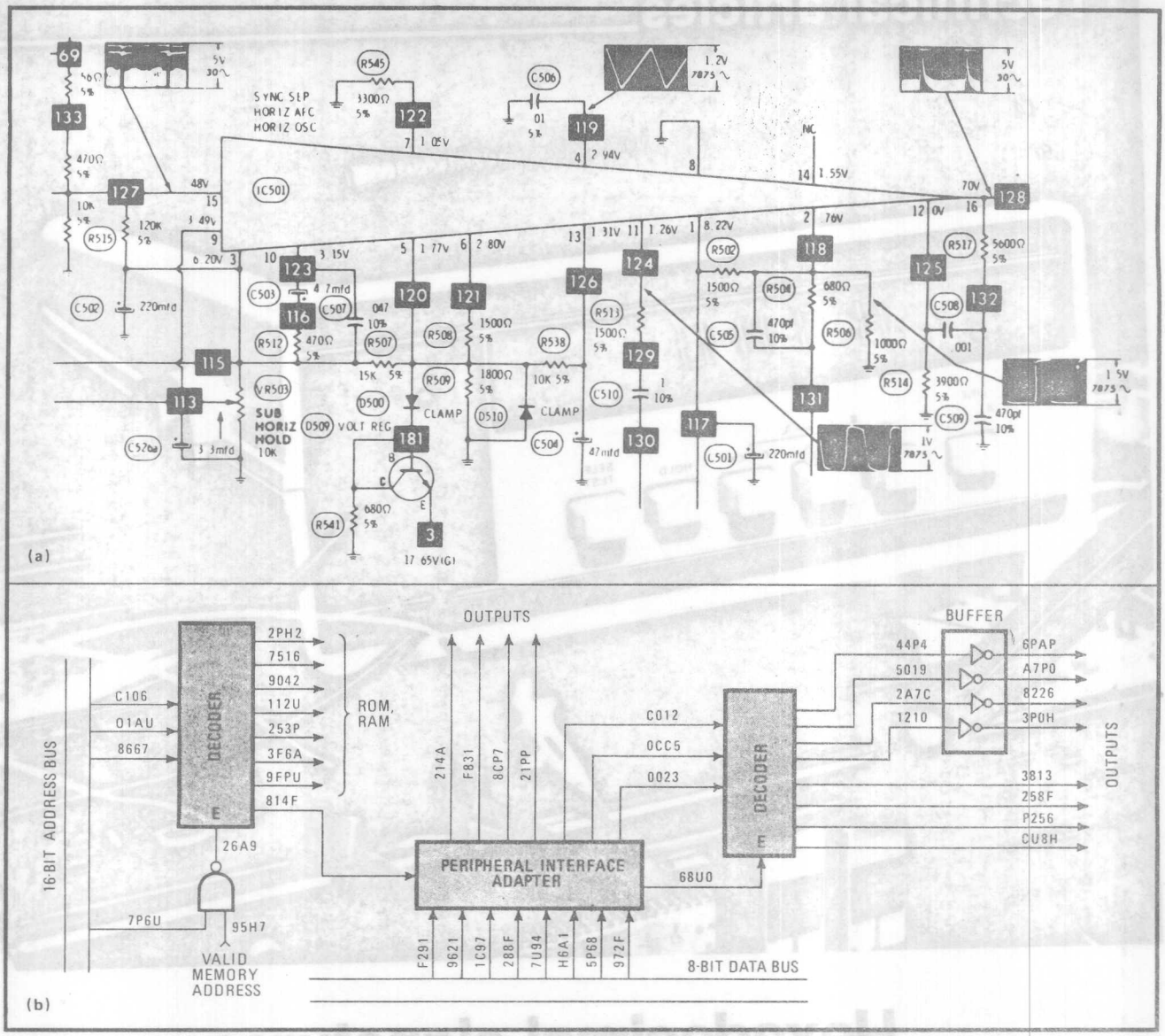
by Gary Gordon and Hans Nadig, *Hewlett-Packard Corp., Santa Clara, Calif.*

☐ The number of microprocessor-based products manufactured each month is approaching the total of all installed computers and minicomputers, raising the question: "How will they be serviced?" Traditional digital servicing, in which defective modules are swapped for good ones, creates substantial inventory and handling costs. A much more economical alternative is a new technique called signature analysis, with which a product can readily be serviced down to the component level.

Signature analysis is based on the time-honored technique of signal tracing. When its requirements are designed into a product, a new test instrument can map lengthy bit streams from the product into short four-digit hexadecimal "signatures." These the technician compares with the correct signatures noted on the system's circuit diagram. If a bit stream is faulty, he traces it back through gates and memory elements until he can isolate an element with correct inputs but faulty outputs. The method has a 99.998% certainty of spotting a faulty bit stream, regardless of its length or the subtlety of its faults.

Signature analysis is more than a new measurement technique. It is a wholly new service philosophy, for the decision whether to adopt it requires a thorough evalua-

Reprinted from **Electronics**, March 3, 1977



1. A new form of digital ID. Diagrams of analog circuits, as for a television set (a), have voltage levels and waveforms noted on them to aid service technicians (circuit is taken from a Howard W. Sams Photofacts service manual). But in digital circuits, all waveforms look alike, and the schematic shown here (b) is annotated instead with hexadecimal signatures that characterize the bit streams unique to each circuit node. A new test instrument derives actual signatures for the technician to compare with the correct ones on the schematic.

tion of its applicability while a product is still in the development stage.

The final decision will usually be based on economics. On the plus side, the technique:

- Increases a product's value to the customer by lowering his cost of ownership.
- Lowers warranty costs for the manufacturer.
- Lowers production-line troubleshooting costs, because it simplifies the job of isolating the speed-related failures, which cannot be caught by present board testers.
- Lowers materials costs, because the product no longer needs to be divided into replaceable modules.

On the minus side, designing signature analysis into a system does add back some other material expenses and also lengthens development time.

On balance, though, the pluses outweighed the minuses in Hewlett-Packard's first applications of the technique to two digital voltmeters. As will be described later, development time and costs rose roughly 1%. But the return on this investment was substantial. Factory costs dropped, since the 1% or so increase in parts cost caused by adding jumpers and extra read-only-memory space was more than offset by the roughly 5% decrease in other materials costs. As a matter of fact, HP has opted for signature analysis for the majority of its upcoming microprocessor-based instruments and other data products.

Analog roots

Signature analysis reverts to the advantages of analog-equipment servicing. There, too, the serviceman uses an annotated schematic (Fig. 1a), which clearly shows voltages and waveforms and allows him to make repairs without a detailed understanding of the circuits. By

How accurate is signature analysis?

For any technique intended to pick up errors in bit streams, it is important to have a measure of the accuracy with which it performs that function. To calculate the accuracy of signature analysis, the first step is to define an error bit stream as a hypothetical sequence related to an erroneous data bit stream in the following way: in the error bit stream, the bit or bits that are in error in the data stream show up as 1s, while bits not in error—regardless of whether they are 1s or 0s in the data stream—show up as 0s. Then, in, say, a 500-bit data sequence, if bit 42 is in error—whether it is a 1 or a 0—bit 42 of the corresponding error bit stream will be a 1 and all 499 other bits will be 0s.

The second step is to invoke the principle of superposition, which is applicable because the feedback shift register is linear. This states that the response of the register to the sum of two inputs is the same as the sum of its responses to the individual inputs. (Note that superposition is used only for this analysis and is not used in the actual instrument.)

From this it follows that if the register input is considered to be the sum (modulo 2) of two sequences—a data bit stream and an error bit stream—then the signature it should display will be equivalent to the sum (modulo 2) of the individual signatures.

Consider this case of summing the two signatures. If there are one or more errors in the data bit stream, the error bit stream will contain 1s in those locations. Then, the sum of their signatures should be different from the signature of the data bit stream itself. It follows that the signature of the error bit stream must be anything but 0 for errors to be detected. This deduction then leads us to examine the conditions under which the signature of the error bit stream becomes 0—the case where errors would go undetected.

With a 16-bit shift register the error bit stream's signature is never 0 for streams of 16 bits or less that contain a 1. This happens because the first 1 to enter the register never has time to leave it before the signature is complete and can never be canceled by a fed-back bit. Thus all errors are caught.

For length 17, one error-bit-stream sequence will be missed; that which starts with a 1, and then has a 1 present at each bit-time where the first 1 is fed back, thus canceling each subsequent error bit. Then, as the 17th bit enters the register, the first and only remaining error bit exits from the register. Thus the signature will be 0 even though there were 1s in the original error bit stream. This means that with 17-bit-sequence inputs, one of the 2^{17} possible combinations may be in error and will not be caught. For length 18, three are missed; for length 19, seven are missed; and so on.

In general, the percentage probability of catching an error in a sequence length m with register length n is:

$$100 - 100[H(m-n)][(2^{m-n}-1)/(2^m-1)]$$

where H is the step function (required to make the function zero for sequences of n or less).

In more useful terms, with register length n equal to 16, the error is always less than 1 in 2^{16}, regardless of m, the length of the input stream (as m gets very large, the error term approaches 2^{-n}). This gives rise to the certainty of 99.998% that an error, if present, can be spotted.

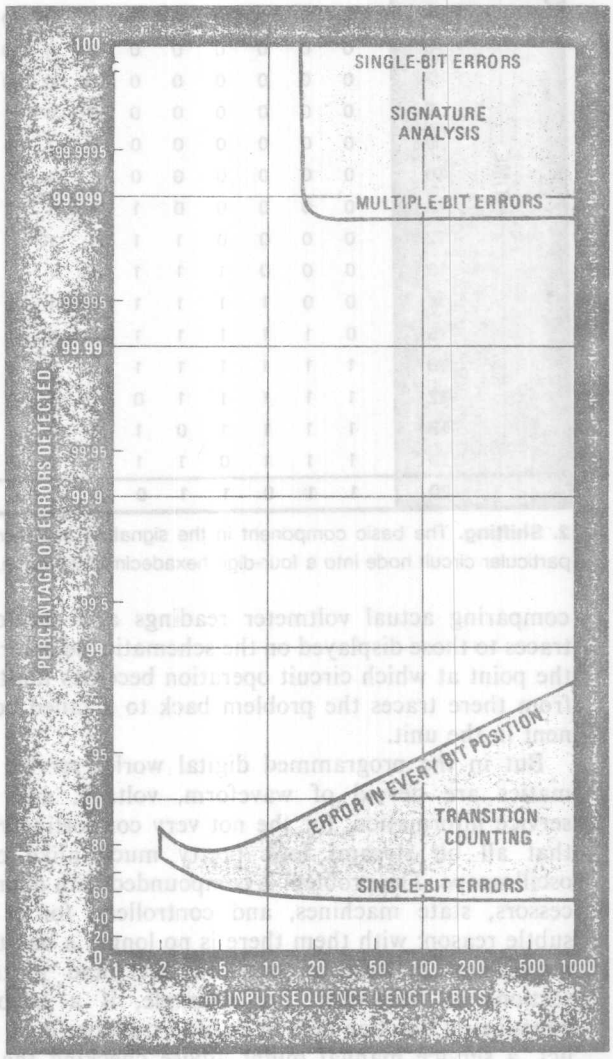
For transition counting, the corresponding probability that it will detect an error in a sequence of length m is expressed by the equation:

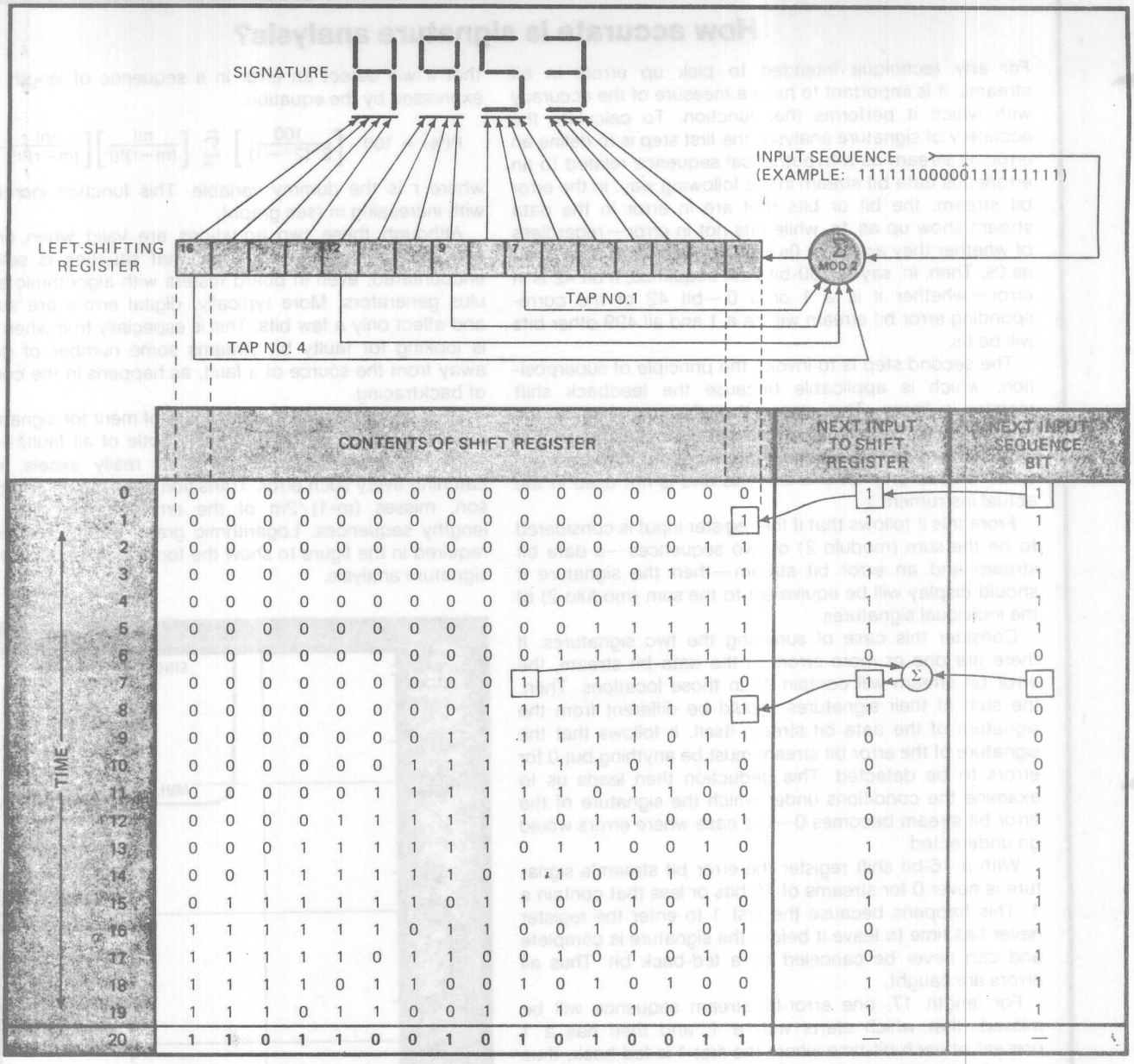
$$P(\%) = 100 - \left[\frac{100}{2^m(2^m-1)}\right] \sum_{r=0}^{m} \left[\frac{m!}{(m-r)!r!}\right]\left[\frac{m!}{(m-r)!r!} - 1\right]$$

where r is the dummy variable. This function increases with increasing m (see graph).

Although these two equations are valid when errors affect the entire bit sequence, that situation is seldom encountered, even in board testers with algorithmic stimulus generators. More typically, digital errors are subtle and affect only a few bits. This is especially true when one is looking for faulty bit streams some number of gates away from the source of a fault, as happens in the course of backtracing.

Thus an equally meaningful figure of merit for signatures is their ability to catch the most subtle of all faults—the single bit error. Signature analysis really excels here, catching every such error. Transition counting, in comparison, misses (m-1)/2m of the errors, nearly 50% for lengthy sequences. Logarithmic graph paper, in fact, is required in the figure to show the formidable superiority of signature analysis.





2. Shifting. The basic component in the signature analyzer is a linear shift register with feedback. This converts the bit stream from a particular circuit node into a four-digit hexadecimal signature. The table shows how a 20-bit input sequence is processed.

comparing actual voltmeter readings and oscilloscope traces to those displayed on the schematic, he determines the point at which circuit operation becomes faulty and from there traces the problem back to a failed component in the unit.

But in the programmed digital world, service schematics are devoid of waveform, voltage, and other service information, for the not very comforting reason that all bit streams look pretty much alike on an oscilloscope. The problem is compounded with microprocessors, state machines, and controllers, for a more subtle reason: with them there is no longer a one-to-one association between product features and particular sections of hardware. For example, if a keyboard-debouncing function fails in an older random-logic product, a service manual might advise checking the integrated circuits that control that function. With microprocessors, on the other hand, key debouncing is more likely a time-shared function tying up the whole processor for a brief moment. When it fails, any one of a large number of ICs could be faulty.

The price of board exchange

As a solution, subdividing the circuitry into replaceable modules has worked reasonably well till now. For one thing, board exchange places relatively few demands on the technical abilities of the serviceman. For another, it is and will remain the fastest way to make repairs when down time is costly, as in large computers and process-control systems. A further advantage is the economy of scale inherent in centralizing repair at the manufacturing site.

But these advantages carry a price. The economy of scale is offset by substantial administrative and inven-

tory costs. A manufacturer may have up to 5% of his assets tied up in service-module inventory, which includes both replacement-board kits and "float" boards in round trip to the factory or waiting in bins. Administrative and handling costs for such a program can also be quite high, particularly as a product approaches obsolescence in the marketplace.

Also, the problem looms of faulty boards in the loop. "Soft" or system-related failures are difficult to detect at repair centers, some of which have reported "no problem found" on 50% of certain returned boards.

Finally, board-level repair is particularly unattractive for supporting products overseas. Turnaround times for modules stretch way out, and import duties of several times the price of the module are often encountered.

For these reasons, centralized board-exchange programs are being widely reevaluated. A partial answer is to move service closer to the customer. For very high-volume products, where board exchange and automated test remain popular, many companies are moving their repair to outlying depots. For lower-volume products where up time is again critical, signature analysis is viable for depot repair of exchanged boards.

Signature analysis was primarily conceived, however, as a more radical change in service strategy. It is a way to substantially reduce repair costs on microprocessor-based products and ROM-based controllers that can stand a few extra hours of down time. Most instruments, computer peripherals, point-of-sale terminals, desk-top calculators, video games, and future citizens' band and television applications fall into this category. The list also includes equipment for which backups are often available: controllers, communications or military equipment, and some of the newer digital products as diverse as taxi-meter and gas-pump controls.

Here the administrative simplicity and cost savings of signature analysis are quite compelling. Troubleshooting by this method requires only a universal $1,000 test instrument, the HP 5004A signature analyzer (see p. 95), and the service manual of the product, which of course must have been specially designed to contain the necessary modest self-test program.

Deriving the signature

Everything depends on the signature. A kind of compressed "fingerprint" of the data present on a node, it is compared with the correct signature printed on the schematic of the product (Fig. 1b) so that any discrepancy may be noted and traced to the source. Clearly, the major figure of merit for any such signature must be the accuracy with which it allows faulty bit streams to be spotted.

HP realized some years ago the potential of the signature-tracing method and investigated numerous ways in which bit streams could be compressed or mapped into signatures. Possibilities are transition counts, 1s counts to generate check sums, and even entropy, the communications measure of information. But a linear-feedback shift register, the method eventually chosen, does a superior job in this regard (see "How accurate is signature analysis?" p. 91).

Linear-feedback shift registers have been used as generators of pseudorandom binary sequences in cryptography, mechanical-vibration control, communications channel testing, and digital radar. But for signature analysis, the register is configured as shown in Fig. 2. Bit sequences being measured are summed in modulo 2 with the register feedback. The register is clocked by the same clock as the bit stream under measurement. Input sequences may be any length, but at the end of the measurement only the residue remaining in the register is looked at. These 16 bits, when displayed in a hexadecimal format, comprise the "signature" of the measured bit stream. A nonstandard hexadecimal character set (0123456789ACFHPU) was chosen for easy readability and compatibility with 7-segment displays.

The table in Fig. 2 shows how the register generates a signature from the 20-bit sequence 11111100000-111111111. Initially (time 0 thru 7) the register acts merely as a shift register. At time 7, the first 1 of the input sequence has reached the first feedback tap (tap 1, Fig. 2). It is fed back and mixed with the input 0, with the result that a 1, not a 0, is next clocked into the register (time 8). This behavior continues until the end of the measurement when a residue of 16 bits, 1101100101010011 (time 20), is all that is left from the 20-bit input sequence. (Note the total dissimilarity in appearance between this residue and the original 11111100000111111111 input sequence.) This residue is displayed in hexadecimal format as H953, the signature of the 20-bit sequence.

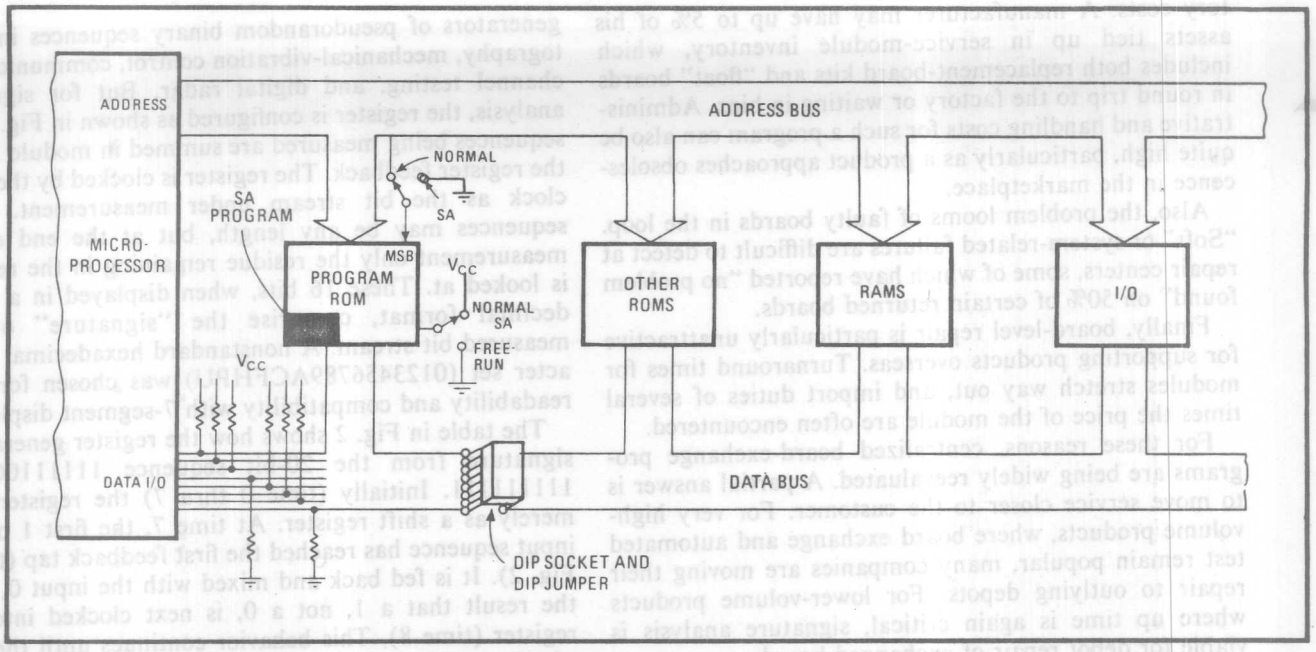
Designing in signature analysis

In an actual circuit designed to incorporate signature analysis, the bit sequences may be any length, probably much longer than 20 bits. A portion of the circuit's read-only memory—perhaps 5%—contains a special program for stimulating the various nodes in the circuit (Fig. 3). The stimulation serves more to "wiggle" or force a state change on the nodes than to generate meaningful data. Frequently this stimulus program may be merged with the product's performance-verification program. The correct signatures are developed by simply exercising the various parts of a circuit that is known to be good and noting the results on the circuit diagram.

A second requirement, beyond stimulating nodes, is to break feedback paths within the circuit either by using hardware switches, jumpers, or connectors, or by disabling gates with software. This requirement is necessary to prevent a fault from being fed back around and perturbing all data nodes. In practice, adding the ability to break feedback paths incurs an incremental hardware expense of less than 1%. (The cost is more than offset, however, by the savings resulting from no longer needing to subdivide the product into small, replaceable modules.) When these two requirements are met, back-tracing a fault to its source is a straightforward process of tracing faulty signatures.

In using signature analysis, the procedure varies slightly depending on whether the fault lies in the kernel (the minimum configuration of microprocessor and ROM necessary to run the simplest test program) or in the outlying circuitry.

If the fault is in the outlying circuitry (a keyboard or



3. Adapting the product. To incorporate signature analysis in a microprocessor-based product, the designer has to add extra program steps to the read-only memory, some switches to put the system into the signature-analysis mode, resistors to force a no-operation instruction when the ROM is disabled, and jumpers that can be removed to isolate portions of the circuit. Here, a 16-pin jumper in a dual in-line package (type AMP 435704-8 or equivalent) is inserted in the data bus.

display scanner, input/output latch, etc.), the field engineer simply switches the circuit to the diagnostic mode. Then, guided by a troubleshooting tree, he uses the test instrument to trace faults back to their source.

But what if the problem lies in the kernel, and even the ROM stimulus program will not run? Here, the microprocessor itself can provide a stimulus if its address counter is allowed to sequence through the address field. To do this it is only necessary to open the data/instruction bus and force the no-operation instruction onto it. This stimulus program checks out all the address lines and the individually enabled ROMs as well. All of these nodes are readily characterized with signatures.

Since signature analysis relies on the ability of a system to control itself in a synchronous manner, asynchronous circuits, like monostables, direct-memory access, dynamic memory, or interrupts, need to be carefully controlled. Generally, simple provisions in the hardware can be made to force them into a synchronous or disabled condition when that is required for a particular test.

The technique in use

As an example, consider the first HP instrument to use signature analysis—the 3455A system voltmeter from the HP Loveland Instrument division in Colorado. The digital portion is quite extensive. It is microprocessor-controlled and contains a self-test program stored in ROM. If the self-test fails, a jumper inside the enclosure is removed, breaking feedback loops and also enabling the signature-analysis routine which is used now to diagnose the instrument.

The decision to go with signature analysis influenced the design in several ways, all of which make it easier to troubleshoot down to the component level. The whole digital portion is on one board. The elimination of connectors and a multitude of smaller pc subassemblies reduced the production cost and also made all the parts easily accessible for testing without the use of special extender boards.

Naturally, some extra design time, a few more ROM locations, and the extra jumper wire were the price paid for this kind of serviceability. The cost evaluation proved to be interesting: the production cost actually fell, and the extra design time amounted to approximately 1% of the overall development time.

Manual aid

Besides the design engineer, the writer of the service manual made an important contribution to the successful application of the signature analysis to the 3455A voltmeter. After learning the internal algorithms of the product almost as well as the designer and having no precedent to fall back upon, he developed a number of innovative ideas for the service approach that were enthusiastically received by the field engineers.

The service manual is written in such a way that a person unfamiliar with the signature analyzer can walk up to a sick voltmeter, read the instructions, and within a short time locate the fault. One element in the manual is a troubleshooting flowchart or tree (Fig. 4), which systematically guides the technician through the fault-finding process.

The initial tests may rely very little on signature analysis, yet may allow isolation of the fault down to a specific area. Diagnostic programs cannot carry on from here, since they do not have access to individual nodes, but it is from here on down to the components that signature analysis excels.

At this level, the repairman uses the annotated sche-

The signature analyzer

The 5004A signature analyzer checks out a compatibly designed digital product by detecting the bit streams at various circuit nodes and displaying them as hexadecimal signatures, which may or may not agree with the correct values noted on the schematic. It is a lightweight, portable instrument, built around the feedback-shift-register circuitry that produces the signatures, and it is equipped with an active probe for data input.

The probe has dual threshold levels that are compatible with transistor-transistor logic. It also serves as a TTL probe, rather like the HP 545A, and in this capacity provides additional troubleshooting information by indicating high, low, bad-level, and pulsing states.

Through an active "pod" on the 5004A's input cable, the product under test supplies the instrument with three gating signals: start, stop, and clock. Start signals the beginning of a measurement window, preparing the shift register to receive information from the data probe. Stop closes the measurement window. Clock is the system clock of the product under test and assures synchronous acceptance of data and gating signals by the shift register. The active edge of each of these gating signals can be selected at the front panel, giving the designer more latitude in applying signature analysis to his product without adding hardware.

The front panel also includes a gate light and an unstable-signature light. The gate light indicates proper start/stop gating operation, remaining on during the measurement window, with stretching to make it visible to the operator during very short on times. The unstable-signature light indicates a difference between signatures in adjacent windows, alerting the user to intermittent faults that may not be apparent from the display.

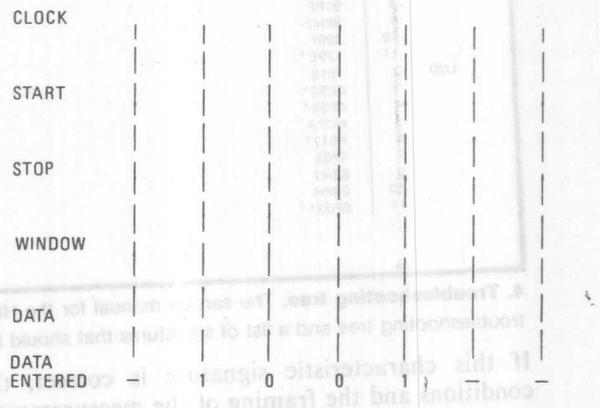
Two useful controls are the hold and reset switches. The hold feature allows observation of single-event (one-shot) signatures, such as a power-on-restart routine. The instrument will display only the signature associated with the first valid window and will hold the display until the probe reset switch is pressed. The hold/reset controls are also useful for taking signatures in awkward locations where it is impossible to simultaneously probe and watch the display.

Since the 5004A is synchronized to the system clock of the product under test (up to 10 megahertz), setup times become important. Setup time is the period during which data must be stable before arrival of the selected clock edge. In the 5004A, the maximum data setup time is 15 nanoseconds (but typically 8 ns). This leaves the balance of the clock cycle for the settling times of components in the product under test. No hold time is required after the selected clock edge.

Any data state changes between clock edges are disregarded. The first data bit accepted during a measurement window is the one coinciding with the synchronized start-signal edge. The last bit accepted is that preceding the synchronized stop-signal edge (see figure).

Tri-state bus architectures are common in many types of equipment and pose the problem of how to interpret the floating state for the purposes of consistent signature detection. Pullup resistors in the circuit under test would force a bus high in the third state, but would slow down the state transitions and possibly cause inconsistent signatures. Instead, the 5004A data probe pulls to the 1.4-volt level, through a 50-kilohm input resistor, and employs hysteresis. This causes the floating state of a tri-state bus to be entered without ambiguity into the signature as the same state (0 or 1) as the most recent valid bit.

To increase the confidence of on-site service, the front panel self-test feature allows a go/no-go checkout of the entire 5004A, including probe, pod, and cables. An internal program exercises the analyzer and displays the result. If this display indicates a malfunction, the 5004A itself can be switched to its own test mode and diagnosed to the component level with another signature analyzer.

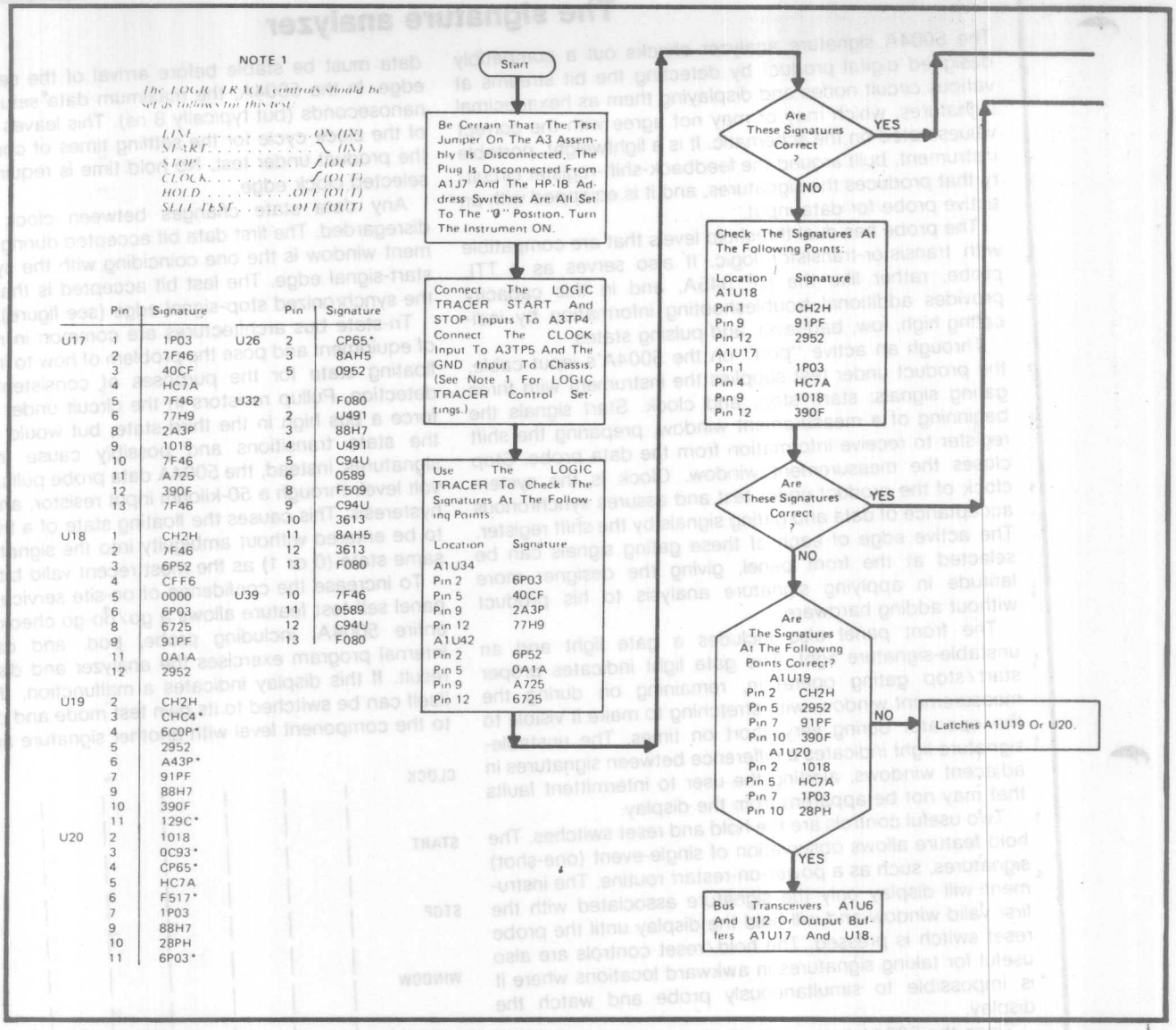


matics and graphs of board layouts, together with the flow chart, to find the bad node. In some cases, the manual includes instructions as to which IC to replace. In other cases the use of a logic probe, which is an integral part of the logic tracer, may be required. A current sensor, such as the HP 547A, helps to find short circuits between traces or to ground and is particularly helpful when bus problems are encountered.

In the case of the voltmeter, the first test checks its kernel, which consists of the microprocessor, the clock circuit, the power supply, and a number of external gates. After the proper functioning of the kernel has been verified, the test setup is changed (one control wire of the logic tracer is moved to another pin in the 3455A), and the remaining portions of the circuit are tested. A special portion of the ROM control loads and reads the random-access memories. Some asynchronous portions require a third test setup. Again, the connection of the start wire is simply moved to the next pin designated for this purpose, and troubleshooting can continue.

Several methods of documentation have been tried successfully. The 3455A service manual shows pictures of the board and defines the setup for each test (Fig. 4). Each picture shows only the signatures related to the particular test, directing the field engineer's effort towards the important areas on the board. The ROM program even simulates interrupt signals, ensuring, however, that they occur predictably at the same spot within a window so that stable signatures result.

The signature analyzer has its own self-check. Each test setup is tested by touching the power-supply voltage with the instrument's probe to input a sequence of all 1s.



	Pin	Signature		Pin	Signature
U17	1	1P03	U26	2	CP65*
	2	7F46		3	8AH5
	3	40CF		5	0952
	4	HC7A			
	5	7F46	U32	1	F080
	6	77H9		2	U491
	8	2A3P		3	88H7
	9	1018		4	U491
	10	7F46		5	C94U
	11	A725		6	0589
	12	390F		8	F509
	13	7F46		9	C94U
				10	3613
U18	1	CH2H		11	8AH5
	2	7F46		12	3613
	3	6P52		13	F080
	4	CFF6			
	5	0000	U39	10	7F46
	6	6P03		11	0589
	8	6725		12	C94U
	9	91PF		13	F080
	11	0A1A			
	12	2952			
U19	2	CH2H			
	3	CHC4*			
	4	6C0P*			
	5	2952			
	6	A43P*			
	7	91PF			
	9	88H7			
	10	390F			
	11	129C*			
U20	2	1018			
	3	0C93*			
	4	CP65*			
	5	HC7A			
	6	F517*			
	7	1P03			
	9	88H7			
	10	28PH			
	11	6P03*			

4. Troubleshooting tree. The service manual for the HP 3455A digital voltmeter, the first instrument to use signature analysis, contains a troubleshooting tree and a list of signatures that should be found at the designated pin numbers of the various devices.

If this characteristic signature is correct, the setup conditions and the framing of the measurement window are verified. Specifically, this tells the user that the switches on the signature analyzer, as well as all the jumpers, switches, and control buttons in front and rear of the voltmeter are correctly set. Thus, the confidence level of the user is very high at the start of a test.

This application of signature analysis went particularly smoothly, because the engineer developing service techniques worked closely with the design engineer. Generally, the design engineer wrote the stimulus routines while the service engineer involved himself with documentation and overall test strategy. This early involvement also ensured that the designer, with the many demands on his time, did not neglect to think about serviceability and, for example, put off allocating read-only-memory space till inconveniently late in the design cycle.

The fact that signature analysis is built into the 3455A voltmeter also made final testing on the production line much easier. A signature analyzer is now a favorite piece of production-line test equipment for the 3455A.

Thus, the signature analyzer promises to have a significant impact on present service costs. With the industry presently spending roughly a billion dollars annually to find the 10 million ICs that fail each year in the field, such a technique is needed. Although IC pretesting and burn-in programs have gone a long way toward weeding out weak devices, changes are nevertheless needed in service strategy as well. The signature analyzer offers a new option for those who are planning service strategies. □

Signature Analysis: A New Digital Field Service Method

In a digital instrument designed for troubleshooting by signature analysis, this method can find the components responsible for well over 99% of all failures, even intermittent ones, without removing circuit boards from the instrument.

by Robert A. Frohwerk

WITH THE ADVENT OF MICROPROCESSORS and highly complex LSI (large-scale integrated) circuits, the engineer troubleshooting digital systems finds himself dealing more with long digital data patterns than with waveforms. As packaging density increases and the use of more LSI circuits leaves fewer test points available, the data streams at the available test points can become very complex. The problem is how to apply some suitable stimulus to the circuit and analyze the resulting data patterns to locate the faulty component so that it can be replaced and the circuit board returned to service.

The search for an optimal troubleshooting algorithm to find failing components on digital circuit boards has taken many directions, but all of the approaches tried have had at least one shortcoming. Some simply do not test a realistic set of input conditions, while others perform well at detecting logical errors and stuck nodes but fail to detect timing-related problems. Test systems capable of detecting one-half to two-thirds of all possible errors occurring in a circuit have been considered quite good. These systems tend to be large, for factory-based use only, and computer-driven, requiring program support and software packets and hardware interfaces for each type of board to be tested. Field troubleshooting, beyond the logic-probe capability to detect stuck nodes, has been virtually neglected in favor of board exchange programs.

The problem seems to be that test systems have too often been an afterthought. The instrument designer leaves the test procedure to a production test engineer, who seeks a general-purpose solution because he lacks the time to handle each case individually.

Obviously it would be better if the instrument designer provided for field troubleshooting in his original design. Who knows a circuit better than its original designer? Who has the greatest insight as to how to test it? And what better time to modify a circuit to accommodate easy testing than before the circuit is in production?

New Tools Needed

But here another problem arises: what do we offer the circuit designer for tools? A truly portable test instrument, since field troubleshooting is our goal, would be a passive device that merely looked at a circuit and told us why it was failing. The tool would provide no stimulus, require little software support, and have accuracy at least as great as that of computer-driven factory-based test systems.

Cover: *Those strange-looking strings of four alphanumeric characters on the instrument's display and the schematic diagram are* signatures, *and the instrument is the 5004A Signature Analyzer, a troubleshooting tool for field repair of digital systems. With a failing system operating in a self-stimulating test mode, the service person probes various test points, looking for incorrect signature displays that can point to faulty components.*

If a tester provides no stimulus, then the circuit under test must be self-stimulating. Whereas this seemed either impossible or at best very expensive in the past, a self-stimulating circuit is not out of the question now. More and more designs are microprocessor-oriented or ROM-driven, so self-stimulus, in the form of read-only memory, is readily available and relatively inexpensive.

By forcing a limitation on software, we have eliminated the capability to stop on the first failure and must use a burst-mode test. Another restriction we will impose is that the device under test must be synchronous, in the sense that at the time the selected clock signal occurs the data is valid; not an unfair condition by any means, and it will be justified in the article beginning on page 15.

There are only a few known methods for compressing the data for a multiple-bit burst into a form that can be handled easily by a portable tester without an undue amount of software. One method used in large systems is transition counting. Another method, a much more efficient data compression technique borrowed from the telecommunications field, is the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator.

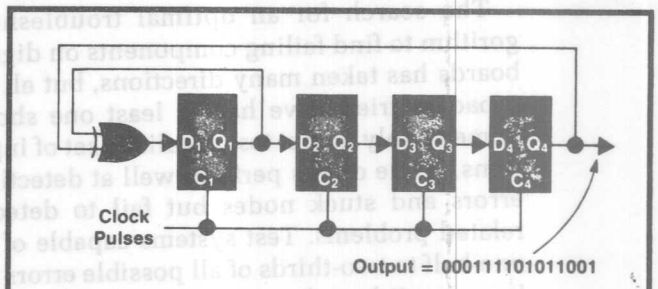
A troubleshooting method and a portable instrument based on this concept turns out to be the answer we are seeking. We call the method signature analysis and the instrument the 5004A Signature Analyzer. The instrument is described in the article on page 9. Here we will present the theory of the method and show that it works, and works very well.

Pseudorandom Binary Sequences

A pseudorandom binary sequence is, as implied, a pattern of binary ones and zeros that appears to be random. However, after some sequence length the pattern repeats. The random-like selection of bits provides nearly ideal statistical characteristics, yet the sequences are usable because of their predictability. A PRBS based upon an n-bit generator may have any length up to 2^n-1 bits before repeating. A generator that repeats after exactly 2^n-1 bits is termed maximal length. Such a generator will produce all possible n-bit sequences, excluding a string of n zeros. As an example, let us take the sequence: 000111101011001. This is a fifteen-bit pattern produced by a four-bit maximal-length generator ($15=2^4-1$). If we were to wrap this sequence around on itself, we would notice that all possible non-zero four-bit patterns occur once and only once, and then the sequence repeats.

To construct a PRBS generator we look to the realm of linear sequential circuits, which is where the simplest generators reside mathematically. Here there exist only two types of operating elements. The first is a modulo-2 adder, also known as an exclusive-OR gate. The other element is a simple D-type flip-flop, which being a memory element behaves merely as a time delay of one clock period. By connecting flip-flops in series we construct a shift-register as in Fig. 1, and by taking the outputs of various flip-flops, exclusive-ORing them, and feeding the result back to the register input, we make it a feedback shift register that will produce a pseudorandom sequence. With properly chosen feedback taps, the sequence will be maximal length. The fifteen-bit sequence above was produced by the generator in Fig. 1, with the flip-flops initially in the 0001 state since the all-zero state is disallowed. The table in Fig. 1 shows the sequence in detail. The list contains each of the sixteen ways of arranging four bits, except four zeros.

If we take the same feedback shift register and provide it with an external input, as in Fig. 2, we can overlay data onto the pseudorandom sequence. The overlaid data disturbs the internal sequence of the generator. If we begin with an initial state of all zeros and supply a data impulse of 1000..., the result is the same sequence as in Fig. 1 delayed by one clock period.



	Cycle	Q_1	Q_2	Q_3	Q_4	$D_1 = Q_1 \oplus Q_4$
Initial State:	0	1	0	0	0	1
	1	1	1	0	0	1
	2	1	1	1	0	1
	3	1	1	1	1	0
	4	0	1	1	1	1
	5	1	0	1	1	0
	6	0	1	0	1	1
	7	1	0	1	0	1
	8	1	1	0	1	0
	9	0	1	1	0	0
	10	0	0	1	1	1
	11	1	0	0	1	0
	12	0	1	0	0	0
	13	0	0	1	0	0
	14	0	0	0	1	1
Begins To Repeat:	15	1	0	0	0	1

Fig. 1. *Signature analysis is a troubleshooting technique that makes use of the cyclic redundancy check (CRC) code, a sort of checksum, produced by a pseudorandom binary sequence (PRBS) generator. Shown here is a feedback shift register that generates a 15-bit PRBS. The outputs of the four flip-flops go through all possible non-zero four-bit patterns and then the sequence repeats.*

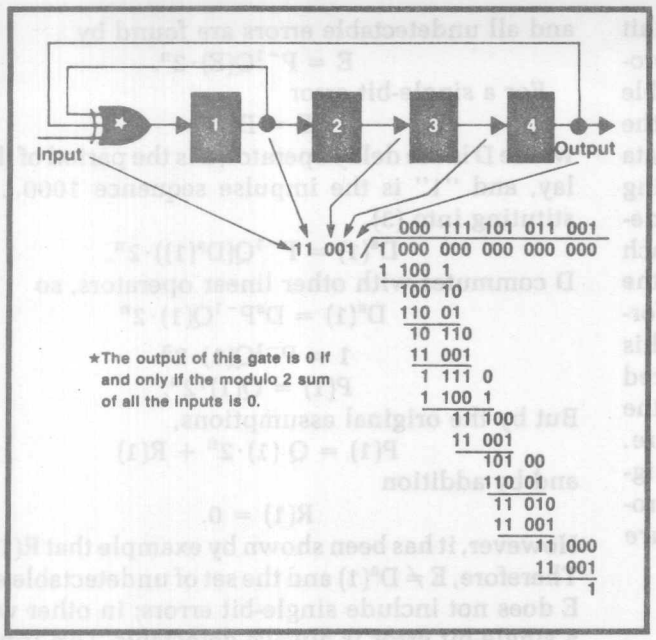


Fig. 2. *When the feedback shift register of Fig. 1 is provided with an external input, data can be overlaid on the PRBS generated by the circuit. Feeding data into a PRBS generator is the same as dividing the data by the characteristic polynomial of the generator.*

Shift Register Mathematics

A shift register may be described using a transform operator, D, defined such that $X(t) = DX(t-1)$. Multiplying by D is equivalent to delaying data by one unit of time. (Recall that we are concerned only about synchronous logic circuits.) In Fig. 2 the data entering the register is the sum of samples taken after one clock period and four clock periods along with the input data itself. Thus, the feedback equation may be written as $D^4X(t) + DX(t) + X(t)$ or simply X^4+X+1.

It happens that feeding a data stream into a PRBS generator is equivalent to dividing the data stream by the characteristic polynomial of the generator. For the particular implementation of the feedback shift register considered here the characteristic polynomial is X^4+X^3+1, which is the reverse of the feedback equation. Fig. 2 shows the register along with longhand division of the impulse data stream (100...). Keep in mind that in modulo-2 arithmetic, addition and subtraction are the same and there is no carry. It can be seen that the quotient is identical to the pattern in Fig. 1 and repeats after fifteen bits (the "1" in the remainder starts the sequence again).

Because the shift register with exclusive-OR feedback is a linear sequential circuit it gives the same weight to each input bit. A nonlinear circuit, on the other hand, would contain such combinatorial devices as AND gates, which are not modulo-2 operators and which would cancel some inputs based upon prior bits. In other words a linear polynomial is one for which $P(X+Y) = P(X) + P(Y)$. Take the example of Fig. 3, where the three different bit streams X, Y, and X+Y are fed to the same PRBS generator. Notice that the output sequences follow the above relationship, that is, $Q(X+Y) = Q(X) + Q(Y)$. Also, notice that Y is a single impulse bit delayed in time with respect to the other sequences and the only difference between X and X+Y is that single bit. Yet, Q(X+Y) looks nothing like Q(X). Indeed, if we stop after entering only twenty bits of the sequences and compare the remainders, or the residues in the shift register, they would be: $R(X+Y) = 0100$, $R(X) = 0111$.

Error Detection by PRBS Generator

Looking at this example in another manner, we can think of X as a valid input data stream and X + Y as an erroneous input with Y being the error sequence. We will prove later that any single-bit error, regardless of when it occurs, will always be detected by stopping the register at any time and comparing the remainder bits (four in this case) with what they should be. This error detection capability is independent of the length of the input sequence. In the example of Fig. 3, R(X+Y) differs from the correct R(X), and the effect of the error remains even though the error has disappeared many clock periods ago.

Let us stop for a moment to recall our original goal. We are searching for a simple data compression algorithm that would be efficient enough to be usable in a field service instrument tester. As such it was to require only minimal hardware and software support.

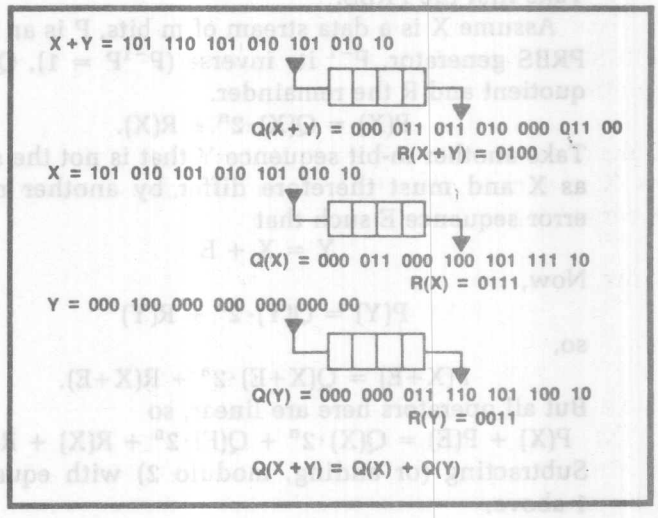


Fig. 3. *Three different input data sequences fed to the same PRBS generator produce very different output sequences even though the input sequences differ by only one bit. If the generators are stopped at some time and the patterns remaining in the flip-flops are compared, they are also different. These remainder patterns are called* signatures. *They show the effects of an error sequence Y added to a data stream X even when the error occurs only once in a long measurement window.*

We have now found such an algorithm. If the circuit designer arranges his synchronous circuit so as to provide clock and gate signals that produce a repeatable cycle for testing, then the feedback shift register is the passive device that we need to accumulate the data from a node in the instrument under test. By tracing through an instrument known to be good, the designer merely annotates his schematic, labeling each test point with the contents of the shift register at the end of the measurement cycle, and uses this information later to analyze a failing circuit. Because this PRBS residue depends on every bit that has entered the generator, it is an identifying characteristic of the data stream. We have chosen to call it a *signature*. The process of annotating schematics with good signatures as an aid in troubleshooting circuits that produce bad signatures has been termed *signature analysis*.

Errors Detected by Signature Analysis

We have claimed that any single-bit error will always be detected by a PRBS generator. But how about multiple errors? Also, our goal was to maintain error detection capability at least as good as existing methods. Earlier mention was made of transition counting, which appears to be the only other method that could easily be made portable. To show how signature analyis stands up against transition counting requires a mathematical discussion of the error detection capabilities of these methods.[4] Take first the PRBS.

Assume X is a data stream of m bits, P is an n-bit PRBS generator, P^{-1} its inverse ($P^{-1}P = 1$), Q is a quotient and R the remainder.

$$P(X) = Q(X)\cdot 2^n + R(X). \quad (1)$$

Take another m-bit sequence Y that is not the same as X and must therefore differ by another m-bit error sequence E such that

$$Y = X + E.$$

Now,

$$P(Y) = Q(Y)\cdot 2^n + R(Y)$$

so,

$$P(X+E) = Q(X+E)\cdot 2^n + R(X+E).$$

But all operators here are linear, so

$$P(X) + P(E) = Q(X)\cdot 2^n + Q(E)\cdot 2^n + R(X) + R(E).$$

Subtracting (or adding, modulo 2) with equation 1 above,

$$P(E) = Q(E)\cdot 2^n + R(E). \quad (2)$$

However, if Y is to contain undetectable errors,

$$R(Y) = R(X).$$

It follows that

$$R(Y) = R(X+E) = R(X) + R(E) = R(X),$$

$$R(E) = 0.$$

Substituting into equation 2,

$$P(E) = Q(E)\cdot 2^n,$$

and all undetectable errors are found by

$$E = P^{-1}Q(E)\cdot 2^n. \quad (3)$$

For a single-bit error

$$E = D^a(1)$$

where D is the delay operator, a is the period of the delay, and "1" is the impulse sequence 1000... Substituting into (3),

$$D^a(1) = P^{-1}Q(D^a(1))\cdot 2^n.$$

D commutes with other linear operators, so

$$D^a(1) = D^aP^{-1}Q(1)\cdot 2^n$$

$$1 = P^{-1}Q(1)\cdot 2^n$$

$$P(1) = Q(1)\cdot 2^n.$$

But by the original assumptions,

$$P(1) = Q(1)\cdot 2^n + R(1)$$

and by addition

$$R(1) = 0.$$

However, it has been shown by example that $R(1) \neq 0$. Therefore, $E \neq D^a(1)$ and the set of undetectable errors E does not include single-bit errors; in other words, a single-bit error is always detectable. (An intuitive argument might conclude that a single-bit error would always be detected because there would never be another error bit to cancel the feedback.)

To examine all undetectable errors as defined by equation 3, it helps to consider a diagrammatical representation, Fig. 4, of:

$$E = P^{-1}Q(E)\cdot 2^n.$$

Since X, Y, and E are all m-bit sequences, it follows that $Q\cdot 2^n$ must be an m-bit sequence containing n final zeros. Q therefore contains (m−n) bits. Hence, there are 2^{m-n} sequences that map into the same residue as the correct sequence, and there are $2^{m-n}-1$ error sequences that are undetectable because they leave the same residue as the correct sequence. 2^m sequences can be generated using m bits and only one of these is correct, so the probability of failing to detect an error by a PRBS is

$$\text{Prob (PRBS, fail)} = \frac{\text{Undetectable Errors}}{\text{Total Errors}} = \frac{2^{m-n}-1}{2^m-1}.$$

For long sequences, large m,

$$\text{Prob (PRBS, fail)} \approx 1/2^n.$$

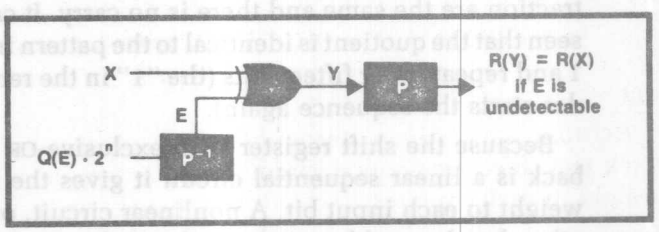


Fig. 4. *A diagrammatical representation of errors undetectable by signature analysis. For long data sequences the probability of not detecting an error approaches $1/2^n$, where n is the number of flip-flops in the feedback shift register.*

In summary, a feedback shift register of length n will detect all errors in data streams of n or fewer bits, because the entire sequence will remain in the register, R(X) = P(X). For data streams of greater than n bits in length, the probability of detecting an error using a PRBS is very near certainty even for generators of modest length. The errors not detected are predictable and can be generated by taking all m-bit sequences with n trailing zeros and acting upon such sequences by the inverse of the n-bit PRBS generator polynomial P, that is

$$E = P^{-1}(Q \cdot 2^n).$$

Furthermore, such error detection methods will always detect a single-bit error regardless of the length of the data stream. It can also be proved that the only undetectable error sequence containing two errors such that the second cancels the effect of the first is produced by separating the two errors by exactly 2^n-1 zeros.[1] The one sequence of length n+1 that contains undetectable errors begins with an error and then contains other errors that cancel each time the original error is fed back.

Errors Detected by Transition Counting

It appears that signature analysis using a PRBS generator is a difficult act to follow, but let us give transition counting a chance. A transition counter assumes an initial state of zero and increments at each clock time for which the present data bit differs from the previous bit. With a transition counter the probability of an undetected error, given that there is some error, is:

$$\text{Prob (Trnsn, Fail)} = N_u/N_t,$$

where N_u = number of undetected errors and N_t = total number of errors. But

$$N_u = \sum_{r=0}^{m} p_{ur}$$

where ρ_{ur} = Prob (undetected errors given r transitions). However,

$$p_{ur} = N_{ur} \cdot p_r$$

where N_{ur} = number of undetected errors given r transitions, and p_r = Prob (counting r transitions). Reducing further,

$$N_{ur} = N_r - N_c,$$
$$p_r = N_r/N_s,$$

where N_c = number of ways of counting correctly (=1), N_s = total number of m-bit sequences, and N_r = number of ways of counting r transitions:

$$N_r = \binom{m}{r} = \frac{m!}{r!(m-r)!}$$

The binomial coefficient $\binom{m}{r}$ expresses the number of ways of selecting from m things r at a time. Looking back to the original denominator,

$$N_t = N_s - N_c.$$

Putting all of this together

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} (N_r - N_c)(N_r \cdot N_s)}{N_s - N_c}.$$

Or,

$$\text{Prob (Trnsn, Fail)} = \frac{\sum_{r=0}^{m} \left[\binom{m}{r} - 1\right] \binom{m}{r}/2^m}{2^m - 1}$$
$$\approx 1/\sqrt{m\pi}.$$

This is the probability of a transition counter's failing to detect an error in an m-bit sequence.

A similar argument finds the probability of the specific case where a single-bit error is not detected by a transition counter. There are 2^m sequences of m bits and any one of the m bits can be altered to produce a single-bit error, so that there are $m \cdot 2^m$ possible single-bit errors. To determine how many undetected single-bit errors exist, we must look at how to generate them.

Upon considering the various ways of generating single-bit errors that are undetectable, a few observations become obvious. We can never alter the final bit of a sequence, because that would change the transition count by plus or minus one, which would be detected. The only time we can alter a bit without getting caught is when a transition is adjacent to a double bit; that is, flipping the center bit in the patterns 001, 011, 100, or 110 will not affect the transition count. In other words, the transition count for ...0X1... and ...1X0... does not depend on the value of X.

Since our transition counter assumes an initial 0 state, the first bit of the sequence, regardless of its state, can be flipped without affecting the transition count, provided that the second bit is a one. In this case only the second of m bits is predetermined, i.e., $b_2 = 1$, and there are 2^{m-1} ways of completing the sequence. Any bit other than the first or last, that is, the m−2 bits from b_2 through b_{m-1}, can be altered without affecting the transition count if the bit in question is flanked by a zero on one side and a one on the other. For a given bit b_i we have free choice of m−1 bits, since as soon as we select b_{i-1} then b_{i+1} is forced to the opposite state. There are $(m-2) \cdot 2^{m-1}$ of these midstream errors. Adding the 2^{m-1} sequences where b_1 can be changed we have a total of $(m-1) \cdot 2^{m-1}$ sequences containing single-bit errors that cannot be detected by a transition counter. But earlier we showed that the total number of single-bit errors was $m \cdot 2^m$, hence the probability of failing to detect a single-bit error is

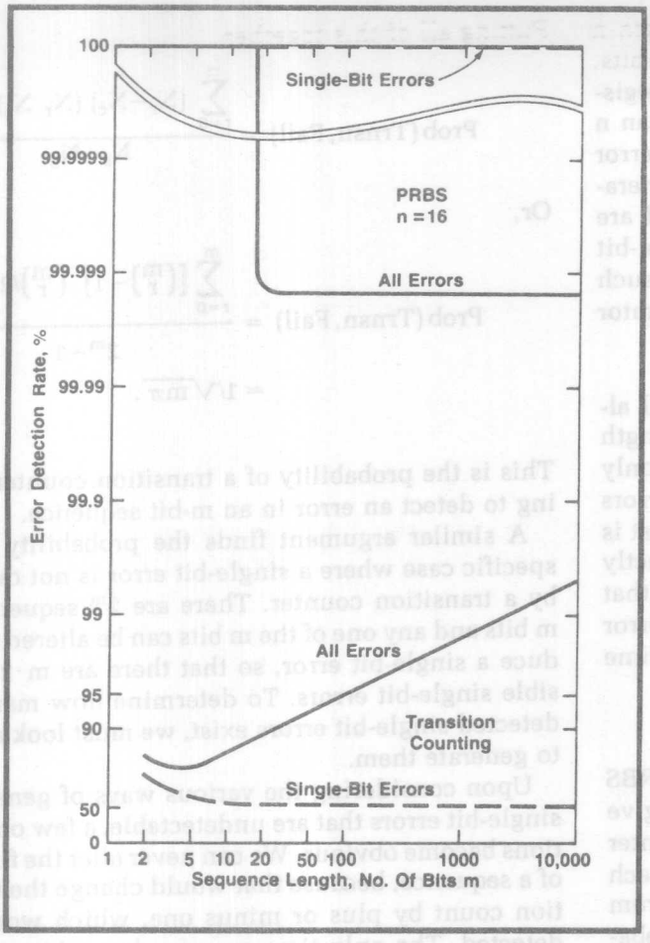


Fig. 5. *Probability of detecting errors for signature analysis and transition counting as a function of the length of the data sequence. n=16 for the PRBS generator.*

$$\text{Prob (Trnsn, Fail, single-bit)} = \frac{(m-1)\cdot 2^{m-1}}{m\cdot 2^{m}} = \frac{m-1}{2m}$$

$$\approx 1/2.$$

It may be noted that the failure rate is actually somewhat higher, because a counter of limited length will overflow for long sequences, leaving some ambiguity. It can be shown that because of this overflow an n-bit transition counter will never detect more than $1/2^n$ of all errors.

Signature Analysis versus Transition Counting

We can now plot the probabilities of detecting any error using a transition counter versus a PRBS generator (see Fig. 5). It is interesting to note that the transition count method looks worst on single-bit errors, exactly where the feedback shift register never fails. Overall the transition counter looks pretty good, detecting at least half of all errors, but even a one-bit shift register could do that. The four-bit PRBS generator used in earlier examples will always detect better than $(100-100/2^4)=93\%$ of all errors. It seems conclusive that the PRBS method puts on a good performance, and if we want it to do better we merely add one more bit to the register to halve the rate of misses.

How Close Do We Want to Get?

We set out to find a means of instrument testing at least as good as present computer-based methods. These existing systems generally perform as well as the engineer who adapts them to the circuit under test. The task of adapting a circuit to be tested by signature analysis is very much the same as adapting to any other tester—engineering errors are assumed constant. If the PRBS technique is used for backtracing to find faulty components in field service, then the largest remaining block of human error is the ability of the service person to recognize a faulty signature.

It seems that a four-character signature is easily recognized, while the incidence of correct pattern recognition falls off with the addition of a fifth character. We tried this on a statistically small sample of people and found it to be so. Electronically, four hexadecimal characters is sixteen bits. A few bits more or less is not likely to complicate a shift register, but it would have an adverse effect on the user. Sixteen bits gives a detector failure rate of less than sixteen parts per million (one in 65,535), adequate for most purposes, so we settled on a four-character signature.

Since the signature offers no diagnostic information

Last In → A	B	C	D ← First In	Display
0	0	0	0	0
1	0	0	0	1
0	1	0	0	2
1	1	0	0	3
0	0	1	0	4
1	0	1	0	5
0	1	1	0	6
1	1	1	0	7
0	0	0	1	8
1	0	0	1	9
0	1	0	1	A
1	1	0	1	C
0	0	1	1	F
1	0	1	1	H
0	1	1	1	P
1	1	1	1	U

Fig. 6. *In the HP 5004A Signature Analyzer, n=16 and the remainder, or signature, is displayed as four non-standard hexadecimal characters. Each character represents the outputs of a group of four flip-flops as shown here.*

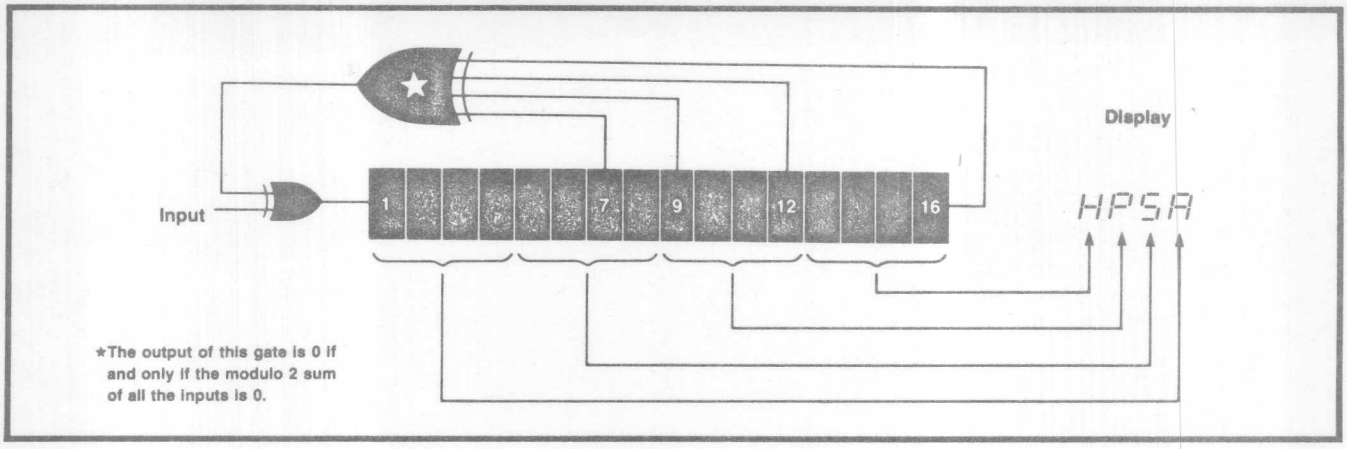


Fig. 7. *The 16-flip-flop PRBS generator used in the 5004A Signature Analyzer.*

whatsoever, but is purely go/no-go, the character set was not restricted, except to be readable. Numbers are quite readable but there are not enough of them. Another consideration was that for an inexpensive tool, seven-segment displays are desirable. The chosen character set (Fig. 6) is easily reproduced by a seven-segment display and the alpha characters are easily distinguishable even when read upside down. A further psychological advantage of this non-standard ("funny hex") character set is that it does not tempt the user to try to translate back to the binary residue in search of diagnostic information.

Register Polynomial

We have decided on a four-character display for a sixteen-bit register, but it remains to select the feedback taps to guarantee a maximal length sequence. It happens that this can be done in any of 2048 ways.[2] The computer industry uses two:

$$\text{CRC-16} = X^{16}+X^{15}+X^2+1,$$

and

$$\text{SDLC(or CCITT-16)} = X^{16}+X^{12}+X^5+1.$$

But each of these is reducible:

$$\text{CRC} = (X+1)(X^{15}+X+1),$$

and

$$\text{SDLC} = (X+1)(X^{15}+X^{14}+X^{13}+X^{12}+X^4+X^3+X^2+X+1).$$

The X+1 factor was included in both to act as a parity check; it means that all undetectable error sequences will have even parity. This is apparent by looking at the original polynomials and noting that they each have an even number of feedback taps, so an even number of error bits is required to cancel an error. For our purposes this clustering of undetectable errors seems undesirable. We would like a polynomial that scatters the missed errors as much as possible. For this reason we would also like to avoid selecting feedback taps that are evenly spaced or four or eight bits apart because the types of instruments, microprocessor-controlled, that we will most frequently be testing tend to repeat patterns at four and eight-bit intervals. The chosen feedback equation is:

$$X^{16}+X^{12}+X^9+X^7+1,$$

which corresponds to the characteristic polynomial

$$P(X) = X^{16}+X^9+X^7+X^4+1.$$

This is an irreducible maximal length generator with taps spaced unevenly (see Fig. 7). Our relatively limited experience with this PRBS generator has shown no problems with regard to the selection of feedback taps. The test of time will tell; even the CRC-16 generator seems to have fallen out of favor with respect to that of SDLC after having served the large-computer industry for well over a decade.

References

1. W.W. Peterson, and E.J. Weldon, Jr., "Error-Correcting Codes," The MIT Press, Cambridge, Massachusetts, 1972.
2. S.W. Golomb, "Shift-Register Sequences," Holden-Day, Inc., San Francisco, 1967.

Robert A. Frohwerk

Bob Frohwerk did the theoretical work on signature analysis. He received his BS degree in engineering from California Institute of Technology in 1970, and joined HP in 1973 with experience as a test engineer for a semiconductor manufacturer and as a test supervisor and quality assurance engineer/manager for an audio tape recorder firm. He's a member of the Audio Engineering Society. Bob was born in Portland, Oregon. Having recently bought a home in Los Altos, California, he and his wife are busy setting up a workshop for Bob's woodworking and metal sculpture, putting in a large organic garden, and planning furniture projects and a solar heating system. Bob also goes in for nature photography, sound systems, and meteorology (he built his wife's weather station).

Signature Analysis in the 5342A

Incorporating microprocessor control into the 5342A Microwave Frequency Counter made it possible to develop a powerful measuring instrument at a substantial reduction in cost. Besides providing many operational benefits, such as keyboard entry of frequency and amplitude offsets, resolution selection, and offset recall, microprocessor control enhances the serviceability of the 5342A by providing powerful diagnostic routines, also selectable from the front-panel keyboard, that aid the service person in fault isolation and instrument verification (see Fig. 1). Other microprocessor routines, exercised every time the instrument is turned on, check the health of ROMs and RAM and display error codes if all is not well.

Despite the diagnostic aids provided by the microprocessor, placing a microcomputer inside a sophisticated measuring instrument also introduces some serviceability problems. After the first prototype was constructed, we discovered it was impossible to isolate certain failures to a particular assembly using traditional troubleshooting equipment and techniques.

Failures involving the microprocessor assembly and the individual assemblies that interface to the microprocessor assembly were extremely difficult to troubleshoot. Even after hours of troubleshooting, it was uncertain whether the fault was a control failure originating on the microprocessor assembly, an interface failure originating on an assembly's interface with the microprocessor, or a failure in some other part of the instrument, causing the measurement algorithm to hang up or branch to an incorrect program segment. We needed a quick way to verify proper operation of the microprocessor control assembly.

Fortunately, there was a solution which, even though the instrument had advanced to the prototype stage, was inexpensive to implement and permitted microprocessor verification and fault isolation to the component level. This technique, called signature analysis, relies on a relatively inexpensive troubleshooting instrument—the 5004A Signature Analyzer.[1]

Signature Analysis

Signature analysis, as implemented in the 5004A Signature Analyzer, employs a data compression technique to reduce long, complex data streams at circuit nodes to four-digit hexadecimal signatures. By taking the signature of a suspected circuit node and comparing it to the correct signature, which is empirically determined and documented in the operating and service manual, proper circuit operation is quickly verified. By probing designated nodes, observing good and bad signatures, and then tracing back along the signal flow from outputs to inputs, the cause of an incorrect signature may be discovered and corrected.

In operation, four signals must be supplied to the signature analyzer. A START signal initiates the measurement window. During this time window, DATA from a circuit node is clocked into the signature analyzer. A CLOCK signal synchronizes the data. A STOP signal terminates the measurement window.

There are two ways to implement signature analysis and meet the requirements just mentioned in a microprocessor-based product: free running and software driven. In the free running method, the microprocessor is forced into an operating mode in which it cycles continuously through its entire address field. START/STOP signals are derived from the address bus lines. In software driven signature analysis, a stimulus program is stored in ROM. The stimulus program generates START/STOP signals and can also write repeatable DATA streams onto the data bus for testing other assemblies connected to the microprocessor. Free running signature analysis has the advantage of not requiring any ROM space for storing the stimulus program. Software driven signature analysis has the advantage of being able to exercise a greater portion of the instrument's circuitry. For thorough testing, both techniques could be implemented in the same instrument.

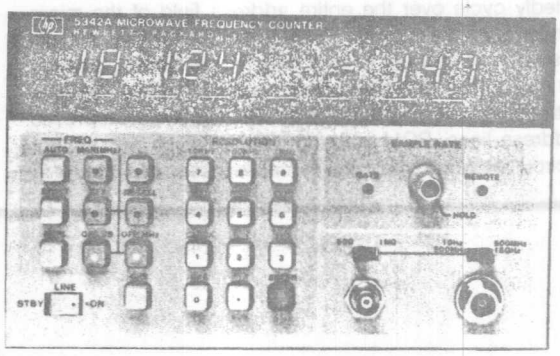


Fig. 1. *Nine diagnostic modes are available with the counter in AUTO mode. The SET key is pushed twice and is followed by the appropriate digit key.*

SET, SET, 0: Indicates that the main synthesizer is sweeping (SP) and that the signal has been placed in the IF (23) and finally that the harmonic determination has been made (Hd). This display is shown in the photograph.

SET, SET, 1: Displays the main synthesizer frequency, the location of the harmonic comb line (e.g., if −, harmonic is below f_x so must add IF result), and the harmonic number N.

SET, SET, 2: Displays results of counter A accumulation during acquisition.

SET, SET, 3: Displays results of counter B accumulation during acquisition.

SET, SET, 4: Displays intermediate frequency being counted.

SET, SET, 5: If Option 002, amplitude measurement, is installed, a single corrected amplitude measurement is made and held.

SET, SET, 6: If Option 002, amplitude measurement is installed, a continuous measure of uncorrected amplitude is displayed.

SET, SET, 7: When the signal is removed from the microwave port, the main synthesizer sweeps over its full range in 100-kHz steps.

SET, SET, 8: This mode is a keyboard check.

In the 5342A, lack of ROM space ruled out the software driven implementation. To implement the free running approach in the 5342A, all that was required was the addition of some switches and pull-up resistors to the microprocessor assembly. Fig. 9 on page 9 shows a block diagram of the 5342A microprocessor assembly. The shaded area contains the components added to the assembly to implement free running signature analysis.

To check out the microprocessor assembly, the microprocessor is forced into a free run mode by opening the data bus switches S1 (this prevents data out of the ROMs from altering the forced free run instruction) and grounding the free

run switch S2. When S2 is grounded, a clear B instruction is presented to the microprocessor data input (clear B was chosen to minimize the number of diodes needed). This causes the B accumulator to be cleared and the address to be incremented by 1. Consequently, the address lines from the microprocessor repeatedly cycle over the entire address field of the microprocessor from 0000 to FFFF. By using the most significant address line as both START and STOP for the 5004A, and one phase of the microprocessor clock as the 5004A CLOCK input, repeatable, stable signatures are obtained for the microprocessor address lines, ROM outputs, device select outputs, and most circuit nodes on the microprocessor assembly. By checking the assembly's outputs for correct signatures (documented in the manual), it is possible to verify with a high degree of confidence that the assembly is functioning properly. If a signature is incorrect, then signatures are checked back along the signal flow paths, from outputs to inputs. When a device is found where the output signature is bad but the input signatures are good, that device is replaced.

Reference

1. A.Y. Chan, "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits," Hewlett-Packard Journal, May 1977.

-Martin Neil

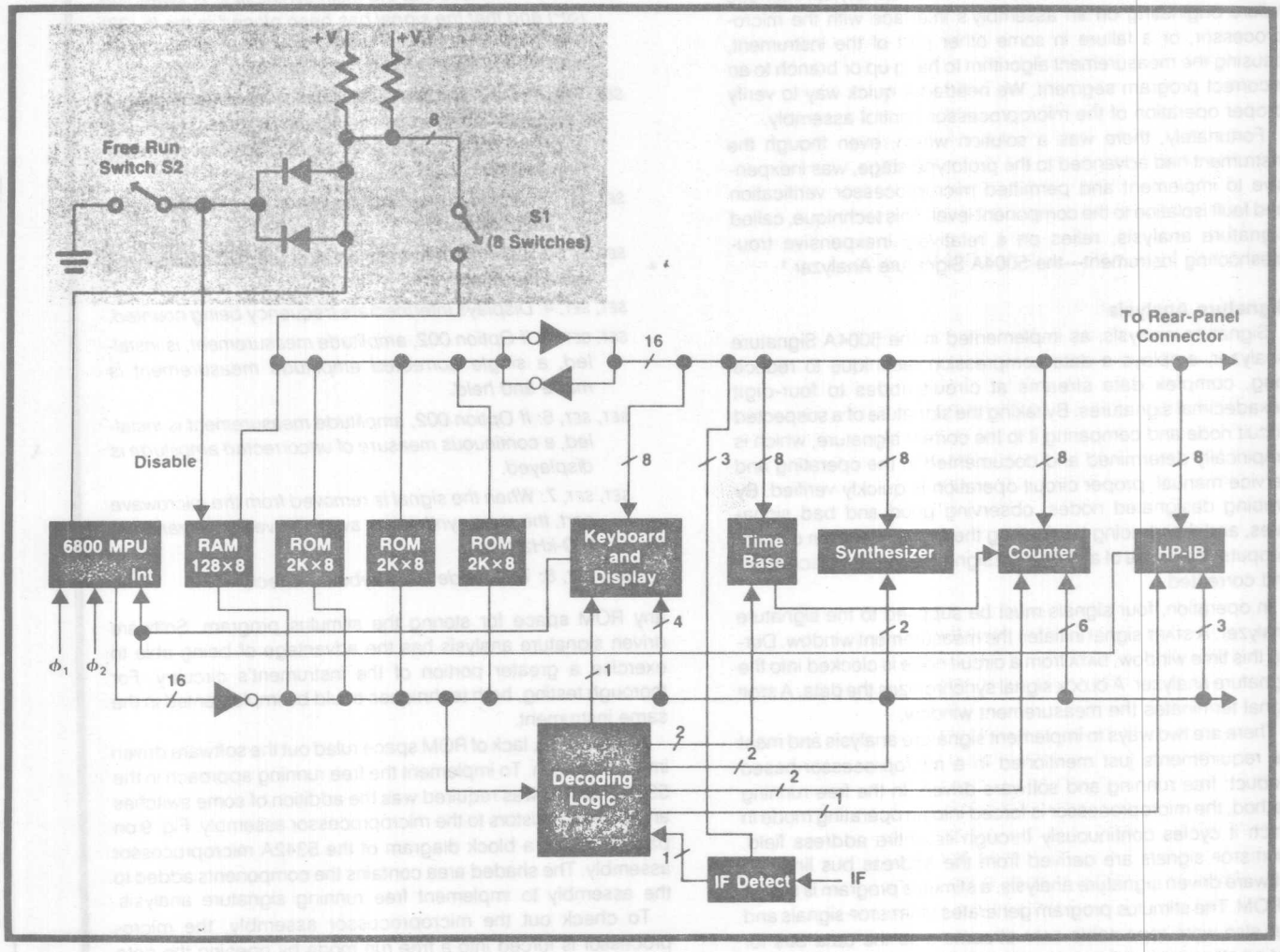


Fig. 9. *5342A microprocessor assembly. Components in the shaded area were added for troubleshooting by signature analysis. Only a few switches and pull-up resistors were required.*

Designing Serviceability into the Model 8568A Spectrum Analyzer

by David D. Sharrit

THE COMPLEXITY OF THE MODEL 8568A Spectrum Analyzer presented several challenges to the serviceability goals set for the instrument. Microcomputer control, the keyboard front panel, the digitally-stored display, and the pilot-signal phase-lock loop are new and very different from previous spectrum analyzers with which production and field repair people are familiar. For this instrument to be serviced and repaired to the component level in a reasonable amount of time, serviceability had to be designed in, beginning with the first prototype.

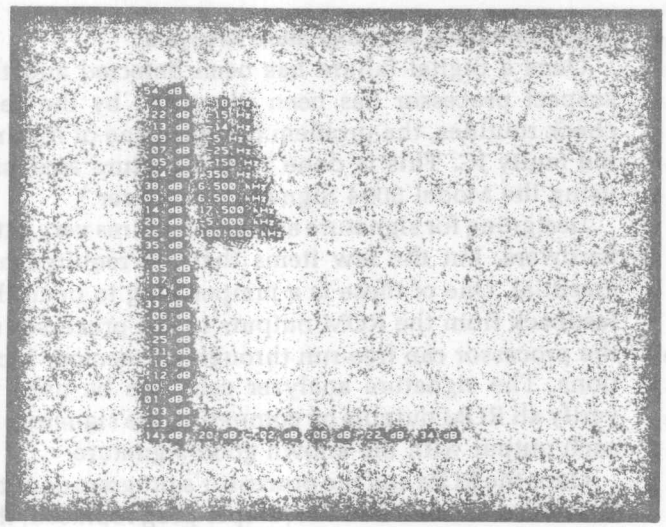


Fig. 1. *Correction factors generated by the error-correction routine can be listed on the CRT to give a check on IF system performance.*

Serviceability was implemented by designing self checks into the digital processors, by taking advantage of the processing power of the analyzer to assist in analog troubleshooting, and by designing for signature-analysis[1] troubleshooting of the digital circuitry.

Troubleshooting from the Front Panel

The system was designed so that most faults occurring in the analyzer's frequency-tuning portions are automatically indicated on the CRT display. Each of three phase-lock loops within the instrument has a lock indicator circuit that generates a flag. If any of these flags indicates an unlock condition when the loop should be locked, the appropriate error message is displayed on the CRT, such as 275 UNLOCK, 249 UNLOCK, and YTO UNLOCK. If either of two counter-locked frequencies controlled by the microprocessor cannot be tuned close enough, then either VTO UNCAL or YTO ERROR is displayed. In most cases, these messages make it possible to isolate the fault to two or three internal assemblies.

The instrument's design is such that normal front-panel operations can be used for quick and accurate troubleshooting of analog portions of the instrument. Because exact center frequencies can be keyed in, because markers can be used to read out frequency, frequency difference, amplitude, and amplitude difference, and also because the internal counter can measure and display the input frequency, such tasks as verifying tuning accuracy and bandwidth accuracy are much simpler than before. In addition, the correction factors generated by the amplitude error-correction routine can be displayed on the CRT, giving the user a quick check on IF section performance (Fig. 1).

Each front-panel key has a secondary function that is accessed with the SHIFT key. For example, pressing the SHIFT key before the FREQUENCY SPAN key is pressed activates the error-correction routine. Several of the shift functions were designed into the analyzer to facilitate troubleshooting. One tells the microprocessor to count and display the actual sweep time, using the internal 10-MHz standard as a clock. It is thus possible to verify the various sweep times without using any external test equipment. This function simultaneously checks the operation of the internal counter.

Other shift functions enable the direct frequency counting of the signal IF, pilot IF, and VTO signals (the VTO generates the frequency offset for the pilot IF path). Another permits direct front-panel control of the digital-to-analog converters (DACs) that normally are controlled by the frequency-tuning algorithms. As an example of how this might be used, the DACs that control the VTO can be set to the end points, 0 and 1023, and the VTO frequency counted at each end point. This enables verification of the VTO oscillator, its tuning range, and the DACs that tune it, all from the front panel.

Making the Unknown Visible

Because of the complexity of the frequency-tuning algorithms, it is not easy for the technician to use the tuning equations to determine many of the internal control settings at a given center frequency and frequency span. One shift function, FREQUENCY DIAGNOSTICS ON, displays most of these settings. These include the tuning DAC settings, the harmonic number, the divide-by-M phase lock numbers, and the calculated frequencies for the VTO and the pilot third LO. Once these numbers are obtained, it is possible to probe the internal circuitry to verify proper operation.

Because the tuning algorithm for the YTO (YIG-tuned first local oscillator) is an iterative process, some failures cause the microprocessor to spend a long time trying to phase lock the YTO before deciding a failure has occurred, in which case it would display an error message and then initiate a sweep. To avoid this time delay during troubleshooting, a shift key function and an internal test point are provided to perform a phase-lock, flag-inhibit function. This tells the microprocessor to ignore errors when tuning the YTO and to sweep as though everything were all right. This essentially corresponds to opening the loop, permitting the analyzer to be tuned during troubleshooting without the microprocessor's continually trying to correct for the YTO tuning failure.

The power of the internal microprocessor can be further enhanced by connecting an external controller such as the Model 9825A Desktop Computer to the analyzer through the HP interface bus. This permits automated testing and alignment procedures (see box).

Troubleshooting the Digital Section

All the troubleshooting aids just described assume that the digital sections are operating properly. The design of the main microprocessor, the display processor, and the HP-IB interface microprocessor permits independent verification and troubleshooting of each. Each has its own test software and can operate independently for testing purposes.

To provide an overall system go/no-go test, two red LEDs were added to the front panel (INSTR CHECK). Whenever the instrument is turned on or the INSTR PRESET key is pressed, the two LEDs are turned on. The main processor then goes through a self check of internal working registers, performs a checksum of the program memory, pattern checks the display memory, and reads the keyboard to verify that there are no stuck keys or stuck I/O lines. If all these checks pass, then the two LEDs are turned off. If a bad bit is detected in the display memory, LED I stays on; if the I/O-keyboard check failed, LED II stays on. Both LEDs' staying on indicates the probability of a failure in the main microprocessor or the program memory.

Grounding an internal test point forces the processing circuits to cycle repeatedly through this test plus an additional RAM test. Monitoring the amount of time spent in each check provides an indication of the particular ROM, RAM, or display storage bit that failed.

When a problem with the display processor or display memory is suspected (LED I remains on), jumpering two test points enables a special set of test ROMs and disables the normal ROMs. A special CRT test pattern is then displayed, verifying the display processor and memory independently from the main microprocessor, the interface circuits, and the interconnect cable.

Troubleshooting with Signature Analysis

Once a digital failure has been detected and the suspect processor has been identified by the self-check routines, the problem becomes one of finding the faulty IC. This is done with signature analysis, using the Model 5004 Signature Analyzer.[2]

Designing for signature analysis requires very little hardware, but the few items that are required are essential. One of these is a jumper plug to open the feedback from the ROM outputs to the processor so the processor can free-run through all memory locations. The signature analyzer can then be used to verify all ROM outputs by comparing the signature at each output with the known good signature.

Once the ROM and the processor's program counter have been verified and the jumper plug replaced, the ROM programs become the stimulus for testing other devices on the processor bus. Several different stimulus programs are stored in ROM. The first program simply outputs various bit patterns on the bus to check the processor and the output bus. The remaining test programs check, usually one at a time, the other circuits on the bus such as the ALUs, RAMs, displays, keyboard scanners, counters, and so on.

Each test program supplies a synchronous, completely defined, repetitive stimulus that checks the functions of each circuit under test. The stimulus program cannot, of course, rely on any feedback from the device being tested or any device not previously verified; otherwise, it would not be possible to guarantee that the stimulus signals are valid. However, once the ALU is verified, it can be used to generate the test pattern for the RAMs and this basic "kernel" grows until the entire system is verified or the original fault found.

For each stimulus program, then, feedback paths need to be opened. This can sometimes be done in software by simply ignoring certain bus outputs, but it often requires grounding test points that have been designed in to permit qualifiers, interrupts, and so on to be inhibited. In addition, all asynchronous timing is either inhibited or tested only at times when it can be guaranteed to appear synchronous.

Inputs to the system from external sources have to be defined and preferably stimulated by the processor. Two special test extender boards were designed to break internal feedback paths and connect processor-controlled outputs to the external inputs for each board. This permits a complete check of the entire board, including the interface circuitry.

The only additional hardware required to properly implement signature analysis was the test points for connecting the start, stop, and clock inputs of the signature analyzer.

Troubleshooting Aids

The signatures are documented on multicolored diagrams that are removable from the service manual. Printed in black on these diagrams are the good signatures for the IC pins. A green verification path shows the output signatures that must be checked to verify that the board is operating properly. When a bad signature is located, information in red indicates what IC pins should be probed next to backtrack the fault to its origin. Printed in red next to the signature of an input pin is the IC number and pin number of the source of that input, so backtracking can be performed without continually referring to the schematic. Also printed on the diagram are the setup requirements for the test, such as the jumpers required to enable the stimulus test program and the start, stop, and clock test points for the signature analyzer.

Reference

1. R.A. Frohwerk, "Signature Analysis: A New Digital Field Service Method," Hewlett-Packard Journal, May 1977.

2. A.Y. Chan, "Easy-to-Use Signature Analyzer Accurately Troubleshoots Complex Logic Circuits," Hewlett-Packard Journal, May 1977.

David D. Sharrit

A native of Phoenix, Arizona, Dave Sharrit worked 3 years on radar signal processing before joining HP in 1973. Initially he worked on the signal-processing circuits for the 8505A Network Analyzer, which earned him two patents, before going to work on the 8568A. Dave has a BSEE degree from Arizona State University (1970) and has done graduate work at Stanford University. In off hours, Dave enjoys outdoor activities, primarily hiking and bicycling.

Team up a μP with signature analysis and ease troubleshooting in the field

Design troubleshooting aids into your microprocessor-based product right at the start, and you can be sure it will be serviceable later on. In fact, one powerful aid is already there—the μP itself. But adding to that an HP-developed technique called signature analysis—which compresses a long data stream into a unique, readily recognizable "signature" of four hex characters (see box)—will enable you to isolate system faults right down to a single node.

With a μP as part of the design, direct connections between the front-panel controls and circuitry often give way to digitally scanned keyboards, processors, ROMs, RAMs, a/d and d/a converters and multiplexed buses. Without the proper tools, then troubleshooting to the component level can become expensive, time-consuming and frustrating. But thanks to the μP, you can add special test routines and functions that take just 5 to 10% of program space, yet verify performance, simplify adjustments and troubleshooting, and detect some internal failures.

For example, in one HP spectrum analyzer, a front-panel function performs a calibration check on the i-f section, and measures the gain and center frequency of all the bandwidth settings. Deviations are displayed on the analyzer's CRT. Another function measures and displays the analyzer's actual sweep time, so the sweep can be checked against the setting. If any of the internal phase-locked loops fall out of lock, the processor automatically displays that condition on the CRT. Other functions permit the internal d/a converters to be programmed directly for testing, internal control settings to be displayed or internal frequencies to be directly counted.

Check the processor

The processor's ability to help verify and troubleshoot extends from the analog to the digital portions of an instrument. When a processor controls an instrument entirely, you must determine whether the processor is functional before attempting to isolate faults in the rest of the instrument. In a multiprocessor-based instrument, isolating the faulty processor is especially crucial.

Consequently, the primary function of a processor's self-test program is to verify the processor and its digital circuitry, or as much of it as possible. Either a self-test routine will activate every time the instrument comes on or an instrument-preset key or test switch will initiate the test routine. The routine generally includes a pattern test of the processor's internal registers, a checksum verification of the program in ROM, a pattern test of the system RAM, a visual test pattern shown on the instrument's display (CRT, LEDs, etc.), and a check of some of the I/O devices. The go/no-go results of self-test are displayed on the front panel, either by LEDs or by a turn-on message on the CRT.

In operation, a preset/power-up line forces the main processor to its reset condition while turning on red check LEDs on the front panel (Fig. 1). The processor then performs a read-write pattern check of its internal accumulators and registers, performs a checksum of each of the 16 program ROMs and a pattern check on the 16 RAMs, executes a read-write check on the display RAM memory through the digital-storage processor and, finally, reads the status of the front-panel keys from the interface board.

If all tests pass, the check LEDs (two in this case) are turned off. Both on indicates a problem with the main processor, ROM or RAM. One LED on indicates a probable problem in the digital-storage processor or its display RAM; the other LED on indicates an unexpected bit from the interface board.

The HPIB processor board has its own self-check routine and front-panel LED. In addition, the digital-storage processor verifies both itself and its RAM (through an internally-selectable self-test pattern), and generates a CRT test pattern, independently of the main processor and the instrument I/O bus. Usually, then, it's possible to both detect a fault and isolate it to one of the three processors or the I/O bus, from the front panel.

Both the front-panel keys and the self-check routine itself can be checked by keeping a key depressed during the main self-test routine. Problems will be detected when the interface board is read. And one check LED will be kept on to ensure that the two check LEDs aren't turned off improperly.

David Sharrit, Development Engineer, Hewlett-Packard, 1400 Fountain Grove Parkway, Santa Rosa, CA 95404

Reprinted from *Electronic Design*, January 4, 1979.

isolate faults. For example, when checking the program ROMs, the routine generates a checksum for each ROM, which enables the processor to tell you which ROM is bad. One way to tell you is to display the faulty IC number on the CRT. But this would require a known-good I/O bus, digital-storage processor and display RAM.

A simpler method requires a minimal amount of known working circuitry: Write the ROM checksum routine so that the time required to execute unambiguously indicates the bad ROM (Fig. 2). Here, checksums are generated for the upper and lower bytes of the first two kwords. If the sums agree with that stored in the first ROM, then the program continues to the next pair of ROMs; if they don't agree, the routine terminates.

Enter signature analysis

To monitor the execution time of this routine, a 4-bit ring counter hangs on the memory address bus, and is cleared and clocked by the main processor using appropriate addresses. Before starting the routine, the processor clears and clocks the counter once, then does it again when the routine is terminated. The time between the rising edges of the first two counter outputs can then be monitored.

A signature analyzer makes a good monitor here, since with a steady ONE at its data probe, the unit generates a unique signature that depends on the number of clocks between the start and stop inputs. You can then reference the signatures to the bad ROM using a fault table (Fig. 3).

For instance, a signature of 6HF5 indicates that U6 is bad; one of UCF4 indicates that all ROMs are good. Bad RAMs are similarly isolated with their own pattern-check routine and another fault table, using the next pair of counter outputs as the start/stop control lines. The complete self-check routine can be forced to repeat continually by grounding a status line into the processor. Not only does this simplify testing, it helps isolate intermittent failures.

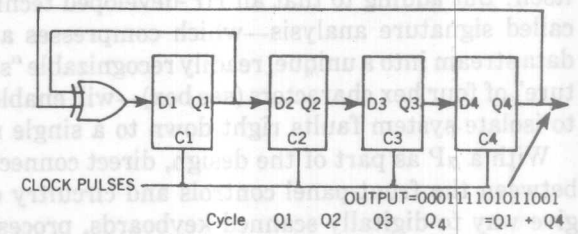
Self-test obviously has its limitations. To run the routine just to check the program ROMs requires that some part of the ROM and a good portion of the processor be operating properly.

If the routine doesn't run, you'll need an additional test, normally called a "free-run" check, to isolate the first ROM, containing the self-test program, from the processor. This is usually done by breaking the feedback path from the ROM back to the processor, thus forcing a NOP instruction, and continually causing the processor to increment the memory address. The actual outputs of the ROM, chip-select lines, and memory address lines can then be verified using the signature analyzer.

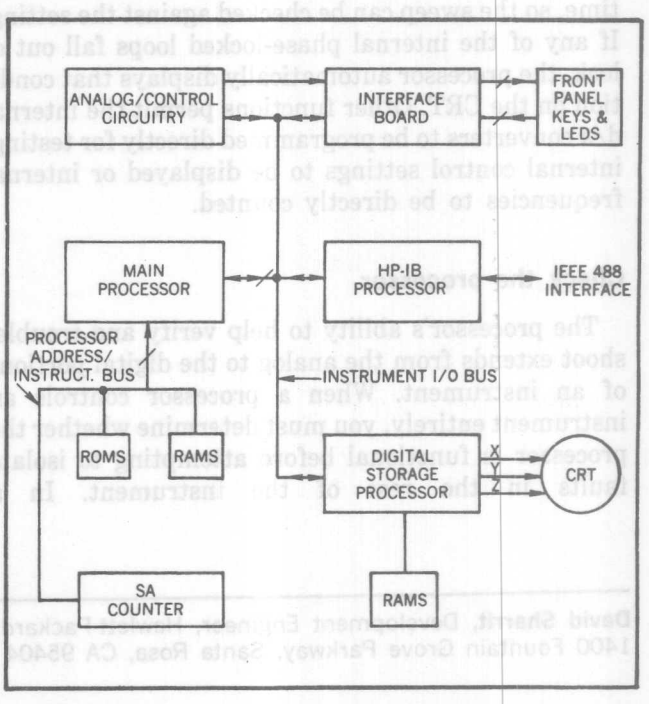
If the processor's memory-output lines don't sequence properly, you may have to do some additional troubleshooting with an oscilloscope to verify the

Revisiting signature analysis

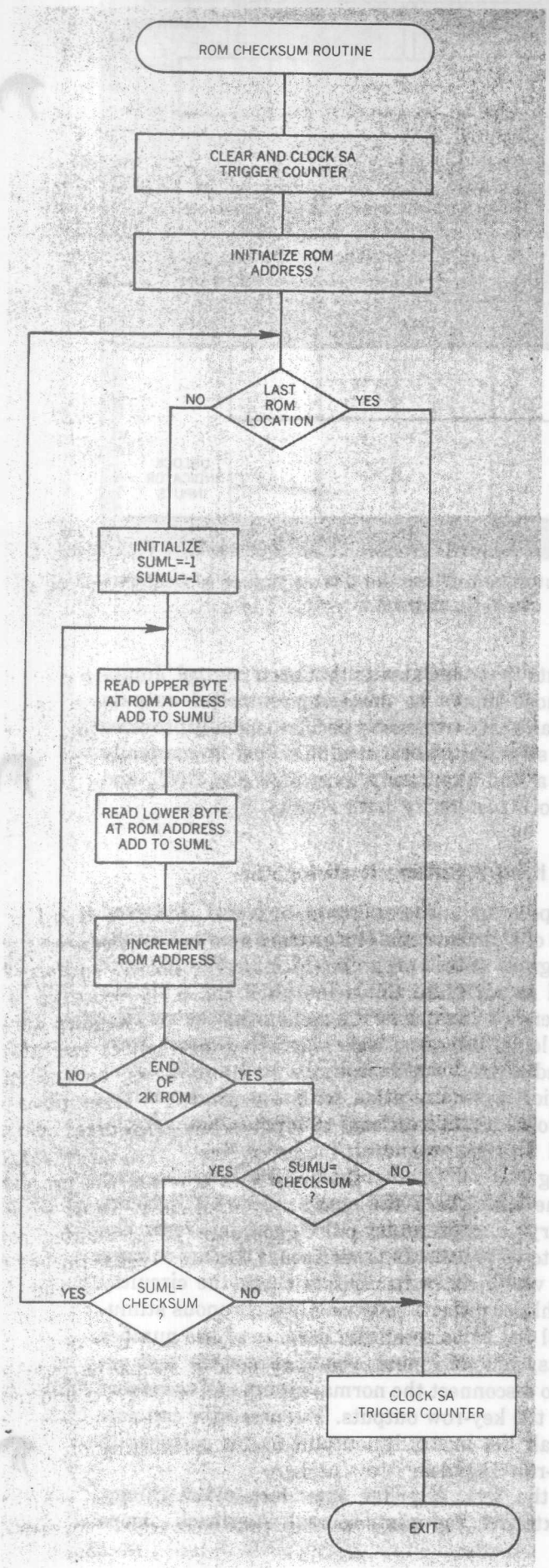
In signature analysis, a cyclic redundancy check code, a sort of checksum, is produced by a pseudorandom binary-sequence generator (see figure). In the figure, a feedback shift register produces a 15-bit sequence: The outputs of the flip flops go through all possible non-ZERO four-bit patterns, and the sequence then repeats. In effect, data at a node are compressed into a form that is handled easily by a portable tester. The compressed data are displayed as four hexadecimal characters—the signature.



Cycle	Q1	Q2	Q3	Q4	=Q1 + Q4
INITIAL STATE: 0	1	0	0	0	1
1	1	1	0	0	1
2	1	1	1	0	1
3	1	1	1	1	0
4	0	1	1	1	1
5	1	0	1	1	0
6	0	1	0	1	1
7	1	0	1	0	1
8	1	1	0	1	0
9	0	1	1	0	0
10	0	0	1	1	1
11	1	0	0	1	0
12	0	1	0	0	0
13	0	0	1	0	0
14	0	0	0	1	1
BEGINS TO REPEAT: 15	1	0	0	0	1



1. **The digital section of a spectrum analyzer** is a likely candidate for self-testing under internal μP control.



2. **A checksum routine verifies** that all program ROMs are OK. Here, the execution time gives away the bad ROM.

processor's clocks, reset and power-supply inputs. If those fall within specifications, you've isolated the fault to the processor chip. If the ROM outputs are good, but the self-check routine still doesn't run when the ROM outputs are reconnected, the processor is faulty.

Other aids are needed

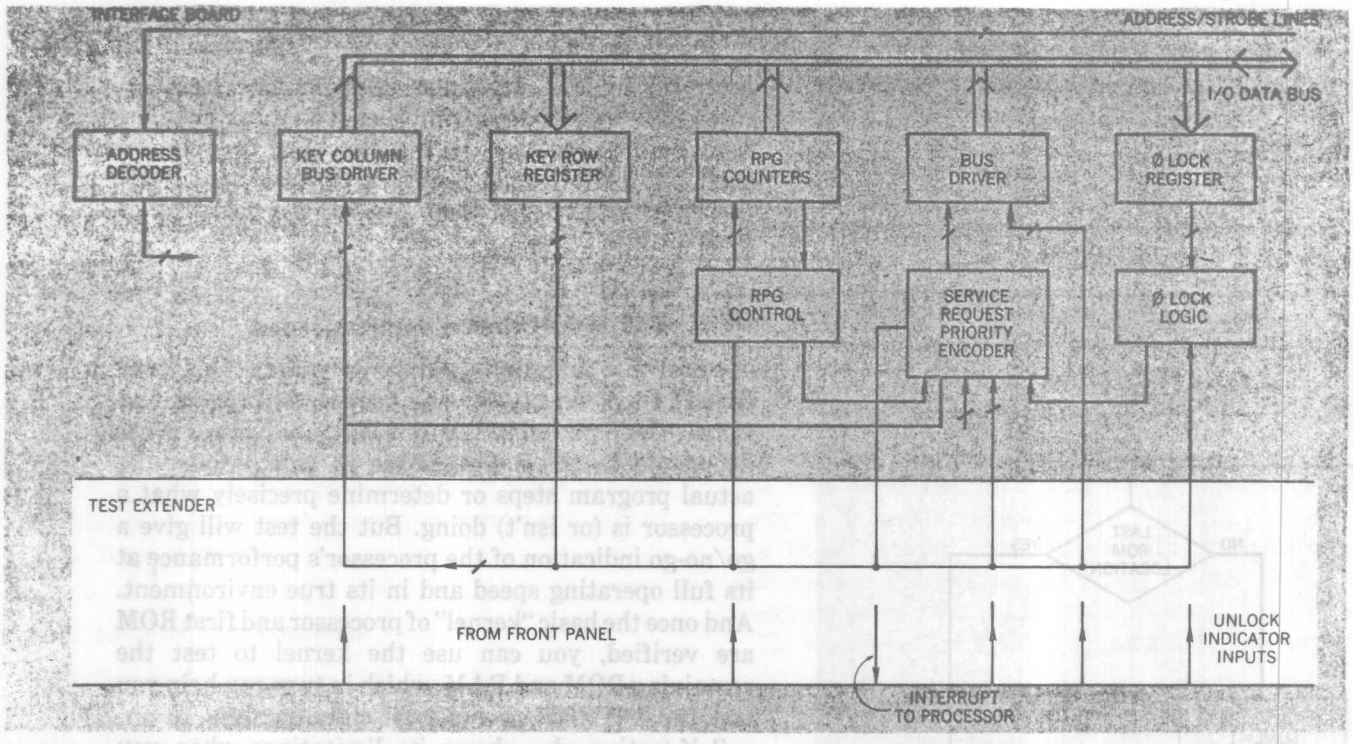
Generally, of course, troubleshooting this way doesn't pin down exactly which processor function has failed. You'll probably call for a logic analyzer or other debug aid in the design phase to troubleshoot the actual program steps or determine precisely what a processor is (or isn't) doing. But the test will give a go/no-go indication of the processor's performance at its full operating speed and in its true environment. And once the basic "kernel" of processor and first ROM are verified, you can use the kernel to test the remaining ROM and RAM, which in turn can help you test the I/O and other digital circuitry, and so on.

Self-testing also shows its limitations when you attempt to isolate faults in the I/O or other circuitry not directly connected to a processor bus. The processor can't always read the outputs of the I/O device —a self-test requirement—though in some cases, it's worth adding the circuitry required to read back the output of an I/O port.

In other situations, the processor can't exercise all

5 VDC SIGNATURE	COURSE OF ACTION
UCF4	ROM IS GOOD, PROCEED TO RAM CHECK
U789	REPLACE HYBRID PROCESSOR A15U13
05C7	U34
095A	U3
0F25	U31
2986	U35
2HP3	U29
31HP	U32
34P5	U33
394U	U5
512U	U2
5PUC	U36
61A0	U7
6HF5	U6
77A0	U4
78FP	U1
CH44	U30
CPU1	U8

3. **Suspect ROMs and RAMs can be spotted** with a look-up fault table, which lists signatures for both bad and good memories.



4. **Mating the technique of signature analysis with** a test program that generates circuit stimulii is a good way to check out circuitry, such as this portion of a spectrum-analyzer interface board.

the inputs required to verify an I/O device thoroughly in its normal configuration. Or it may be able to verify a section of circuitry but not easily pinpoint the faulty device. Is the input buffer bad, or the output buffer? A good way to find out is to team up the processor again with a signature analyzer.

A routine in the processor's program generates the stimulii for the various I/O peripherals; the analyzer verifies the logic circuitry and traces bad signals to their starting point. The stimulus program usually differs from the self-test program because the stimulii must not change because a device is good or bad.

Besides designing the stimulus program, you may have to modify the I/O to accommodate a test configuration. Here are two tips:

1. Either avoid asynchronous timing signals, or disable or ignore them during test modes.

2. Disable feedback paths, such as interrupts to the processor or circuit under test, since a failure anywhere in the loop will make all logic signals appear bad and make isolation impossible. Disabling is easiest when software performs the feedback functions, since a software "path" is easily opened.

To disable a hardware feedback, generally you can add a strategic test point to be grounded during testing, or a jumper to be removed. Or you can connect an unused processor-controlled output to a point in the loop that allows the feedback path to be disabled. Proper board partitioning is as always helpful in troubleshooting, since some feedback paths can be broken and bus-loading problems isolated simply by removing certain boards.

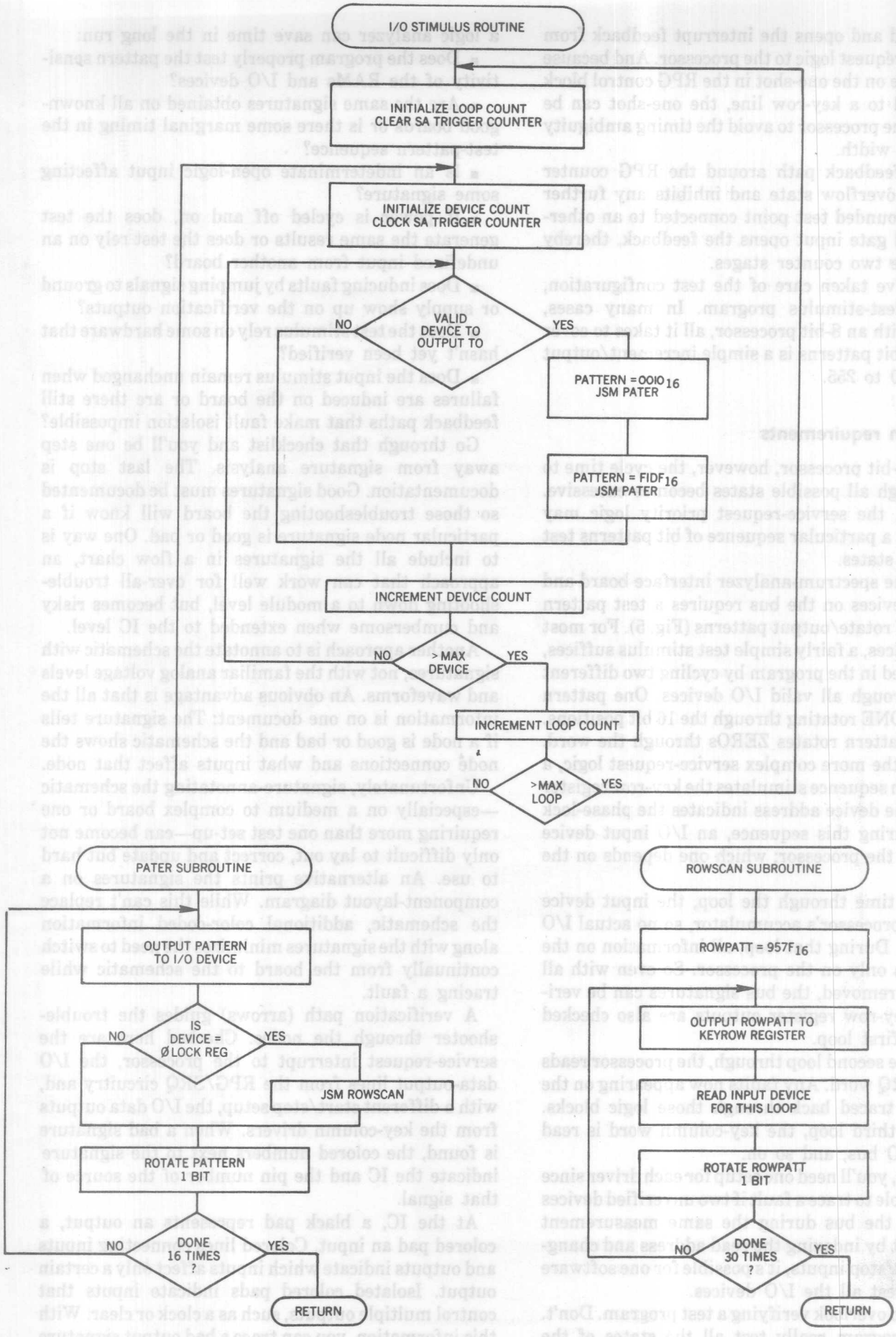
You should also provide some way to stimulate those I/O inputs that aren't under processor control. Jumpering those inputs to processor-controlled outputs (which can be independently verified) usually provides the best stimulus—yet it may be enough to manually connect a line alternately to ground and ONE, and verify both results.

Putting it all together

The spectrum-analyzer interface board is a good example of signature analysis coupled with a stimulus test program (Fig. 4). The board contains the logic required to interface the main processor with the instrument's front panel. It also handles the service-request logic, including lock/unlock indicators, key-down indication, HPIB requests, and so on. Communication with the processor takes place via the bidirectional 16-bit data bus, an address bus, a strobe line and an interrupt line.

During the self-check routine, the processor can check the board only minimally. To do so, it clears the rotary-pulse-generator (RPG) counter, reads the outputs to verify all ZEROs and reads the key-column lines to verify all highs. But to check the circuitry thoroughly requires a repetitive, synchronous stimulus to all the various inputs. Because of the number of inputs, you may need a special-purpose test extender to disconnect the normal inputs and reconnect them to the key-row outputs. The processor can now control all the inputs by outputting bit patterns to the key-row register.

And the test extender also does other things. The extender breaks several feedback loops



5. **A stimulus program** puts an I/O bus (and connected devices) through its paces with different patterns.

on the board and opens the interrupt feedback from the service-request logic to the processor. And because the clear line on the one-shot in the RPG control block is connected to a key-row line, the one-shot can be cleared by the processor to avoid the timing ambiguity of its pulse width.

Another feedback path around the RPG counter detects the overflow state and inhibits any further clocks. A grounded test point connected to an otherwise unused gate input opens the feedback, thereby isolating the two counter stages.

Once you've taken care of the test configuration, write the test-stimulus program. In many cases, especially with an 8-bit processor, all it takes to cover all possible bit patterns is a simple increment/output loop, from 0 to 255.

Test pattern requirements

With a 16-bit processor, however, the cycle time to count through all possible states becomes excessive. In addition, the service-request priority logic may require that a particular sequence of bit patterns test all possible states.

Testing the spectrum-analyzer interface board and the other devices on the bus requires a test pattern with several rotate/output patterns (Fig. 5). For most I/O-bus devices, a fairly simple test stimulus suffices, one generated in the program by cycling two different patterns through all valid I/O devices. One pattern is simply a ONE rotating through the 16 bit positions. The other pattern rotates ZEROs through the word.

To check the more complex service-request logic, a third pattern sequence stimulates the key-row register whenever the device address indicates the phase-lock register. During this sequence, an I/O input device is read into the processor; which one depends on the loop count.

The first time through the loop, the input device acts as the processor's accumulator, so no actual I/O read occurs. During that loop, all information on the bus depends only on the processor. So even with all I/O devices removed, the bus signatures can be verified. The key-row register outputs are also checked during the first loop.

During the second loop through, the processor reads the RPG/SRQ word. Any faults now appearing on the bus can be traced back through those logic blocks. During the third loop, the key-column word is read onto the I/O bus, and so on.

Generally, you'll need one setup for each driver since it isn't possible to trace a fault if two unverified devices are driving the bus during the same measurement window. But by indexing the read address and changing the start/stop inputs, it's possible for one software routine to test all the I/O devices.

It's easy to overlook verifying a test program. Don't. Does the program really test all the states of the priority encoder? Taking the time to question the capability of critical sections of the test program with a logic analyzer can save time in the long run:

- Does the program properly test the pattern sensitivity of the RAMs and I/O devices?
- Are the same signatures obtained on all known-good boards or is there some marginal timing in the test-pattern sequence?
- Is an indeterminate open-logic input affecting some signature?
- If power is cycled off and on, does the test generate the same results or does the test rely on an undefined input from another board?
- Does inducing faults by jumping signals to ground or supply show up on the verification outputs?
- Does the test stimulus rely on some hardware that hasn't yet been verified?
- Does the input stimulus remain unchanged when failures are induced on the board or are there still feedback paths that make fault isolation impossible?

Go through that checklist and you'll be one step away from signature analysis. The last stop is documentation. Good signatures must be documented so those troubleshooting the board will know if a particular node signature is good or bad. One way is to include all the signatures in a flow chart, an approach that can work well for over-all troubleshooting down to a module level, but becomes risky and cumbersome when extended to the IC level.

Another approach is to annotate the schematic with signatures, not with the familiar analog voltage levels and waveforms. An obvious advantage is that all the information is on one document: The signature tells if a node is good or bad and the schematic shows the node connections and what inputs affect that node.

Unfortunately, signature-annotating the schematic —especially on a medium to complex board or one requiring more than one test set-up—can become not only difficult to lay out, correct and update but hard to use. An alternative prints the signatures on a component-layout diagram. While this can't replace the schematic, additional color-coded information along with the signatures minimizes the need to switch continually from the board to the schematic while tracing a fault.

A verification path (arrows) guides the troubleshooter through the nodes. Checked here are the service-request interrupt to the processor, the I/O data-output lines from the RPG/SRQ circuitry and, with a different start/stop setup, the I/O data outputs from the key-column drivers. When a bad signature is found, the colored numbers next to the signature indicate the IC and the pin number of the source of that signal.

At the IC, a black pad represents an output, a colored pad an input. Colored lines connecting inputs and outputs indicate which inputs affect only a certain output. Isolated colored pads indicate inputs that control multiple outputs, such as a clock or clear. With this information, you can trace a bad output signature to its point of origin—where the inputs to a device are good but the output signature is incorrect. ■■

Designing a serviceman's needs into microprocessor-based systems

Mapping tells him where to start; signature analysis locates circuit nodes in trouble; diagnostics check operation

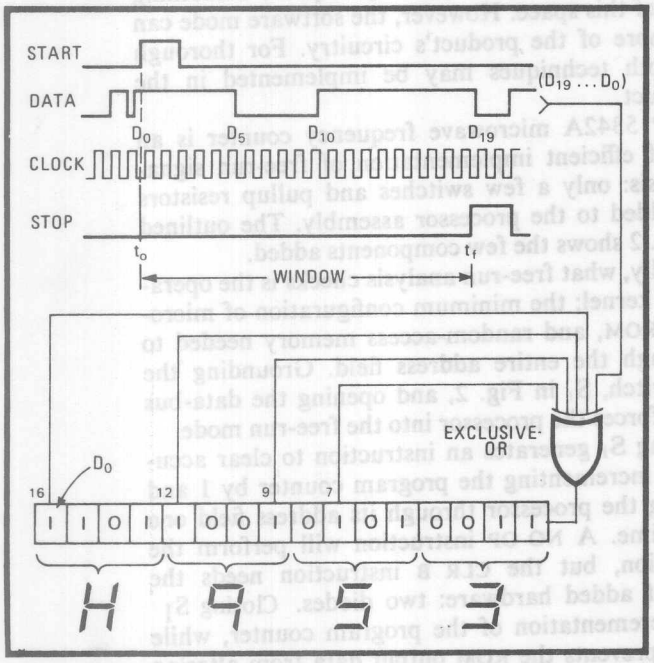
by Martin Neil and Randy Goodner, *Hewlett-Packard Co., Santa Clara Division, Santa Clara, Calif.*

☐ Mention the word "microprocessor" to a serviceman, and watch his face cloud over. He knows that the many advantages the chips offer to both manufacturer and user are coupled with new problems and challenges from his point of view.

There is a way to ease his job, and it is the familiar tack of designing serviceability into the product. However, many designers are unfamiliar with the special servicing requirements of microprocessor-based products.

Troubleshooting woes

Several characteristics associated with such designs present troubleshooting problems for traditional test equipment. One of these is that characterization of the circuitry is difficult because processor firmware often replaces hardware and its operation may be hidden in the software algorithm. A related problem is the dynamic operation of these products, where signals often are active for a few microseconds and then disappear. In a microprocessor-controlled keyboard, for example, checking for signal faults requires knowing when to look as well as where to look.



1. Signature analyzed. This 4-place hexadecimal signature is a compression of the 20-bit data stream entering the linear-feedback shift register. The unique character set, consisting of 0 . . . 9, 4, C, F, P, and U, allows use of a seven-segment display.

A third problem is that the bidirectional nature of the processor bus makes interpretations of address and data information very difficult. Compounding this is the buses's parallel structure, which has many devices ORed together, making for tedious fault detection. Furthermore, the test gear must contend with many operations, since newer instruments may have the processor going through a few thousand steps in a measurement cycle where earlier products usually required fewer than 100 operations.

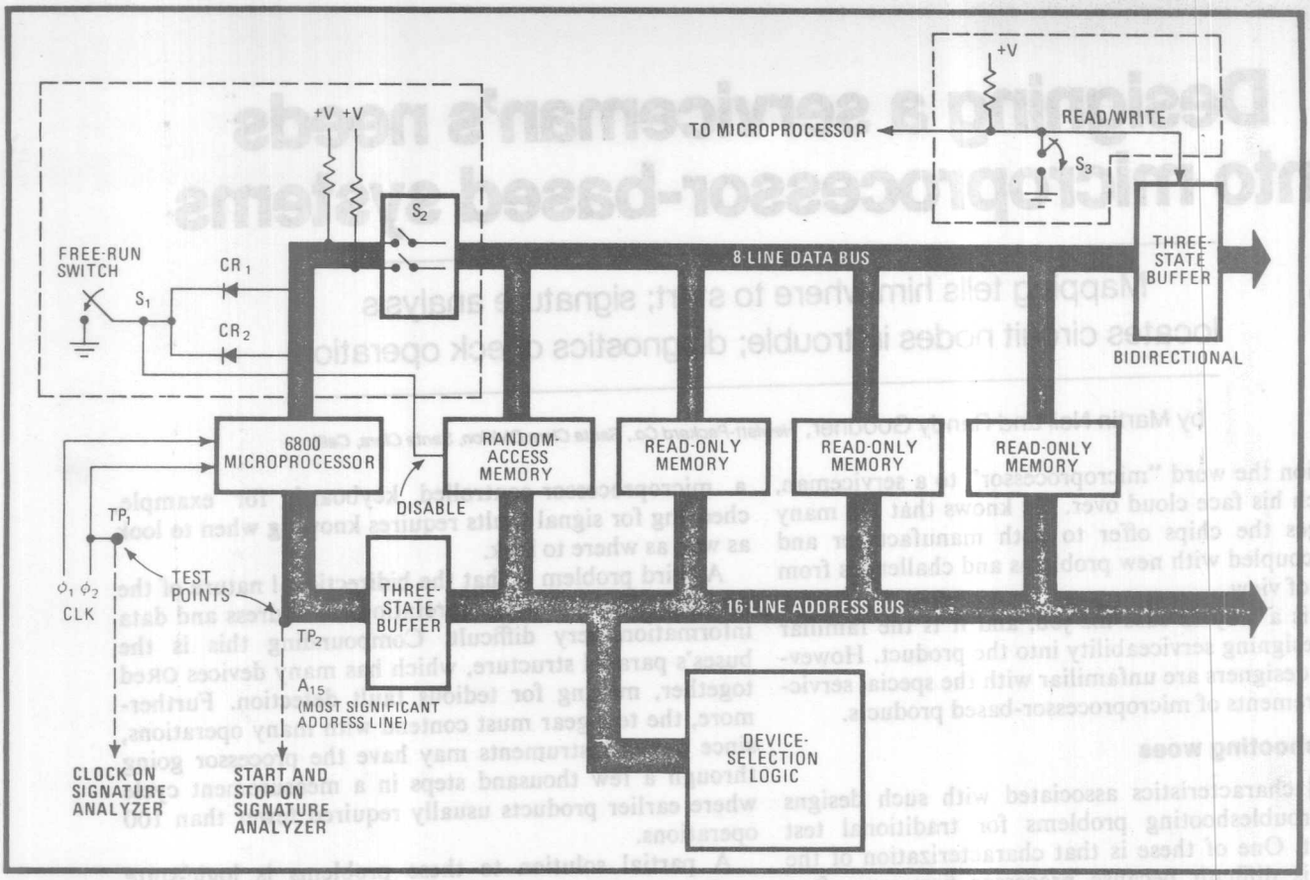
A partial solution to these problems is logic-state analyzers, which help trace the microprocessor's operating algorithm by following the sequence of machine states. Once the state in which the fault first appears is located, traditional test equipment comes into play in order to trace through the nested circuits seeking the component or components giving rise to the fault. However, this procedure can take much time and invariably requires a highly skilled serviceman.

Failures involving the microprocessor assembly and those assemblies that interface to the processor are extremely difficult to troubleshoot. In fact, it may be impossible to isolate these failures with traditional troubleshooting equipment and techniques. Hours of investigation may not determine whether the fault is a control failure originating in the microprocessor assembly, an interface failure, or a failure in some other assembly causing the measurement algorithms to hang up or branch to an incorrect program segment.

To chase the clouds away from the serviceman's face, the designers of microprocessor-based products must take a new look at designing in serviceability. They should look closely at three techniques that can overcome these troubleshooting problems: signature analysis, built-in diagnostics, and mapping. What follows is a discussion of these three, coupled with examples of how specific designs help the serviceman.

Signature analysis

Perhaps the greatest serviceability boon a designer can offer to the product's user is the capability to work with such troubleshooting tools as the HP 5004A signature analyzer. A relatively new technique, signature analysis



2. Open the loop. To implement free-running signature analysis, it is necessary to open the microprocessor assembly's feedback loop to prevent alterations to the free-run instruction. This can be done on the board itself by simply adding a few components.

uses data compression to reduce complex, serial data-stream patterns of any length at a circuit node to a unique four-digit hexadecimal signature [*Electronics*, March 3, 1977, p. 89].

Comparing the signature of a suspect circuit node with the empirically determined correct signature in the service manual will quickly verify circuit operation. The cause of an incorrect signature may be quickly discovered by probing designated nodes, observing good and bad signatures, and tracing the signal flow.

Besides the data from the circuit node, the 5004A needs three signals. A start signal initiates the measurement window during which data is clocked into a 16-bit linear-feedback shift register in the analyzer. A clock signal generated by the unit under test synchronizes the data with the signal analyzer. A stop signal terminates the measurement window, and the 16-bit residue in the shift register appears on the display as a 4-place hexadecimal signature (Fig. 1).

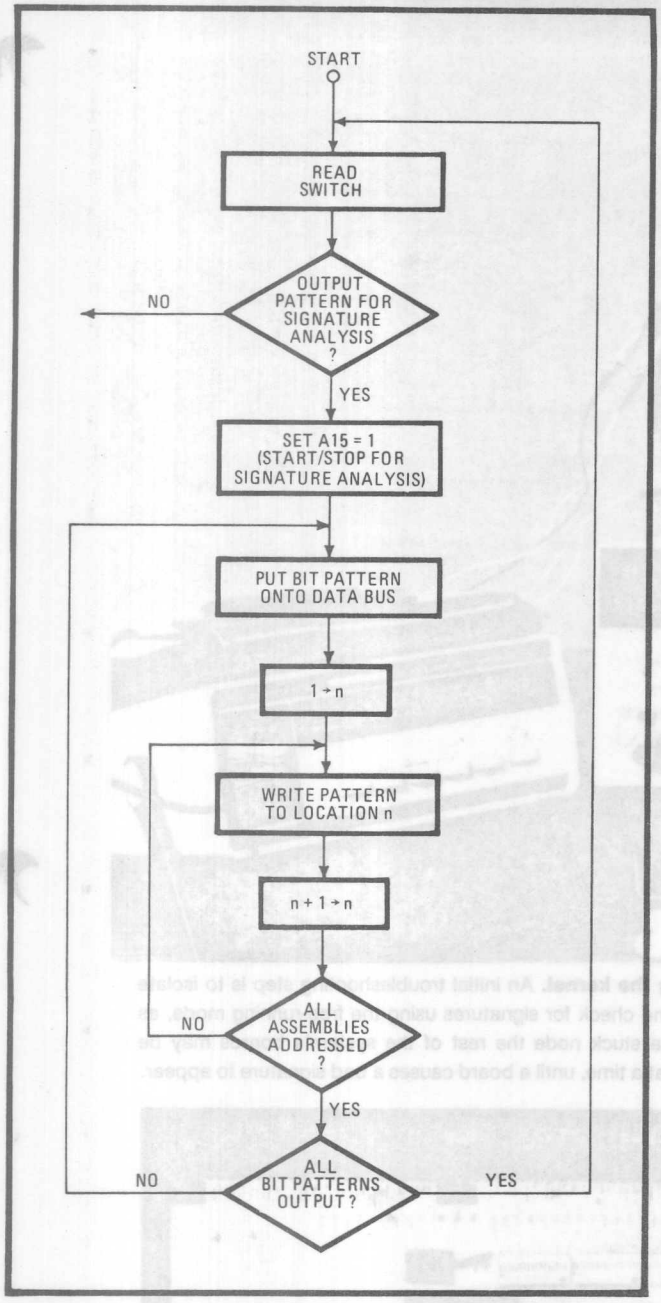
The free-running and software-driven modes are the two basic implementations of signature analysis in a microprocessor-based product. Free-running analysis forces the processor to cycle continuously through its entire address field with start/stop signals derived from the lines of its address bus. Software-drive analysis uses a stimulus program stored in read-only memory to generate the start/stop signals and to write repeatable data streams onto the data bus in order to test assemblies connected to the processor.

The free-running mode occupies no ROM space, while the software-driven mode typically requires something under 5% of this space. However, the software mode can exercise more of the product's circuitry. For thorough testing, both techniques may be implemented in the same product.

The HP 5342A microwave frequency counter is an example of efficient implementation of free-run signature analysis: only a few switches and pullup resistors need be added to the processor assembly. The outlined area in Fig. 2 shows the few components added.

Essentially, what free-run analysis checks is the operation of the kernel: the minimum configuration of microprocessor, ROM, and random-access memory needed to cycle through the entire address field. Grounding the free-run switch, S_1 in Fig. 2, and opening the data-bus switch, S_2, forces the processor into the free-run mode.

Grounding S_1 generates an instruction to clear accumulator B, incrementing the program counter by 1 and thus cycling the processor through its address field one step at a time. A NO OP instruction will perform the same function, but the CLR B instruction needs the minimum of added hardware: two diodes. Closing S_1 allows incrementation of the program counter, while opening S_2 prevents the ROM output data from altering the free-run instruction. Consequently, the processor's address lines cycle repeatedly over the entire address field from 0000 to FFFF. Using the most significant address line (A_{15}) as start/stop signals and one phase of



3. Program-driven analysis. This program stored within a ROM generates the output patterns used to test the system in the program-driven signature-analysis mode. The most significant address bit A_{15} is set high to generate its start/stop signals.

the processor's clock as the clock signal will give repeatable, stable signatures for the address lines, ROM outputs and device-select outputs.

Switch S_3 enables the write buffer so that free-run signatures may be observed there. The read portion of the buffer cannot be checked with free-run analysis; it requires a logic pulser and logic probe.

Software-driven analysis

Exercising a special pattern stored in ROM will provide a more thorough signature analysis than will the free-run mode. This software-driven mode provides a pattern for the boards that are outside the kernel and therefore have no way of generating patterns. It allows troubleshooting of the input registers of devices accepting data from the microprocessor, and will often provide signatures for the circuitry responding to data in those registers.

A typical 8-bit output test pattern would start off with all 0s, go to all 1s, then continue with a walking 1 pattern (10000000, 01000000, . . . 00000001). The processor places the first byte of the pattern on the data bus and instructs a board to accept the byte in a specific location or register. For example, if the board has 12 locations, the processor will write the byte at all 12.

Once done, the processor sends the same byte to locations on the next board. After all locations on all boards have been loaded, the processor places the next byte on the data bus, and the process repeats.

One way to generate start/stop signals is setting the most significant address bit high at the beginning of the test program. With the test program in the lower half of the memory, the most significant address line, A_{15}, can be artificially toggeled at the beginning of the cycle simply by addressing a dummy location in the upper half of the memory. Figure 3 is a flow chart of the program for output pattern generation for this situation.

Program-driven signature analysis can also test inputs from boards that are not part of the kernel. An internal service switch can set the microprocessor to generate an input exercise. Such a test allows these boards to put latched data from every storage location onto the data bus for checking by signature analysis.

There are two primary points of concern in performing such a check of the read operation: determining by sequential elimination which board among many is malfunctioning and insuring that the circuits are properly initialized. Indeed, these concerns apply to both types of program-driven analysis and to free-run analysis as well. Moreover, they are part of a design checklist for designing in signature-analysis capability.

Design points to remember

Experience in designing microprocessor-based products that will work with signature analyzers suggests a list of six considerations that must be part of the design process. These six are:

- Distinguish the kernel.
- Provide a way to open local feedback paths.
- Insure initialization of the boards under test.
- Use address decoding to isolate ROM failures.
- Stabilize the signatures of three-state devices.
- Carefully document signatures.

The kernel should be on its own board, for one of the first steps in troubleshooting a malfunctioning system is to find out if the kernel free-runs. This test cannot be accomplished unless the kernel can be completely isolated (Fig. 4). If a separate board is not possible, then switches must be provided to isolate the kernel's components from the rest of the system.

A good case in point is a malfunction in which some device on the microprocessor's addresses and data buses continuously pulls a line high or low. It is not clear which device is malfunctioning. The solution is to attach to the kernel an extender board modified with isolating switches for the address and data lines (Fig. 5).

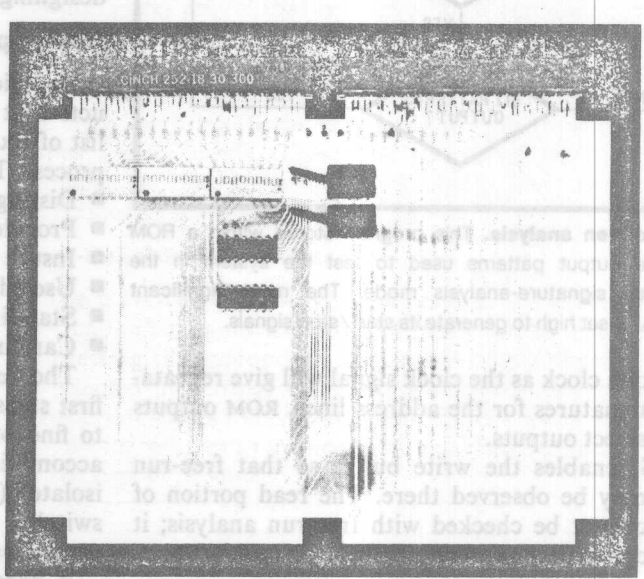
4. Shooting the kernel. An initial troubleshooting step is to isolate the kernel and check for signatures using the free-running mode, as shown. For a stuck node the rest of the system's boards may be added, one at a time, until a board causes a bad signature to appear.

To find the origin of stuck nodes on a product with an extender attached, the serviceman takes the signatures with all bus line switches closed. Then, on the lines giving incorrect signatures, he opens the extender-board switches and takes the signatures on their kernel side. A good signature on this side means that the bad signature on the other side is caused by a bad external device pulling that line high or low.

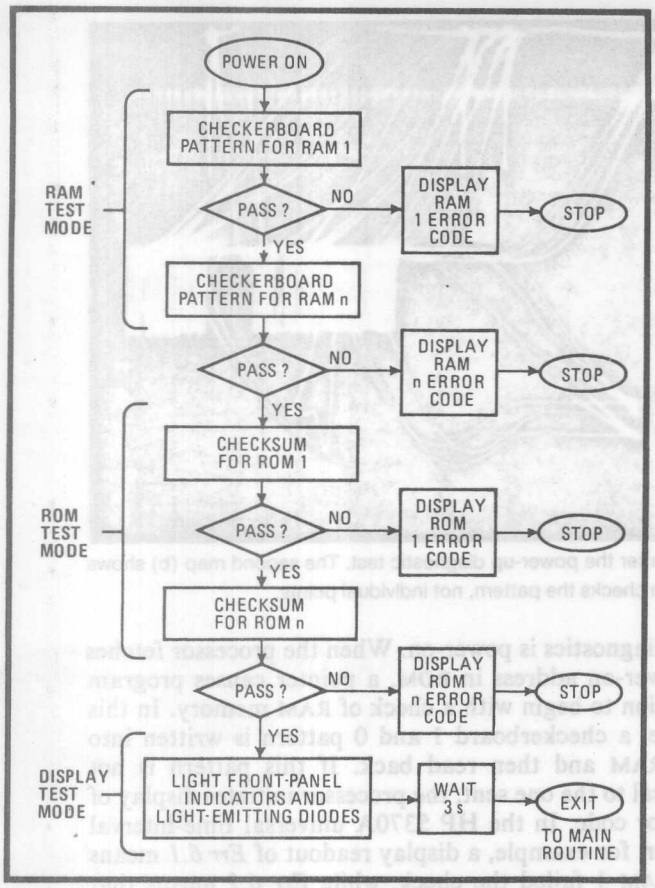
The next step is to determine which device is the bad one, and this can be a much simpler process if the designer puts each subsystem interfaced with the processor on its own board. Then the serviceman can simply add boards to the free-running kernel until the bad signature is seen again. He continues to use the extender board's switches to isolate stuck nodes.

A similar problem can occur with program-driven signature analysis that generates input tests. A read cycle involves all boards putting data on the bus, so it is necessary to test them in sequence by adding them one at a time to the kernel.

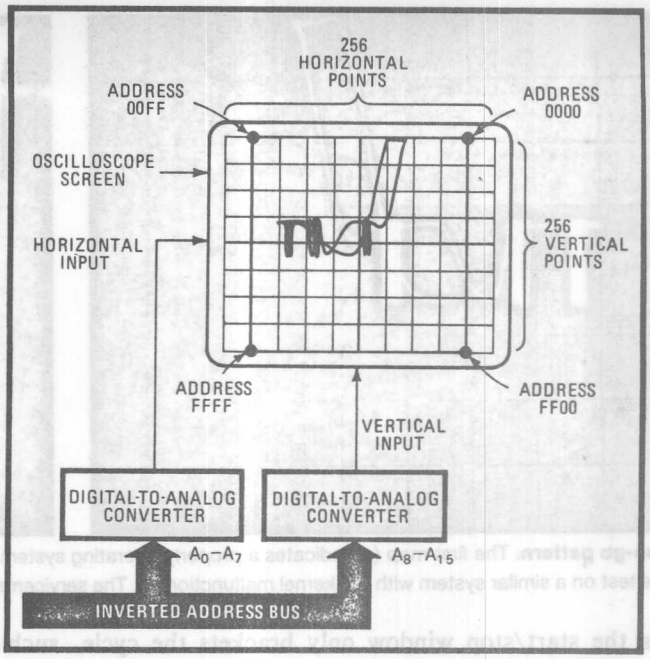
In this case, determining whether a particular board sends out data properly requires its isolation from other boards performing the same function. One tack is to pull all other boards putting out data, excluding the processor and ROM boards. Another tack is to isolate the board under test by an extender card with switches on the data lines. However, opening these switches does more than



5. Isolation. The modified extender board isolates the questionable boards' address lines and data lines from those of the bus. It also provides the serviceman with a manual reset capability and circuitry for address decoding (located near the board's center).



6. Power-on test. The diagnostic program charted describes a test routine for checking a system's RAMs, ROMs, and LEDs. The program is self-initiating at power-on and displays failure error codes. Internal switches or keyboard inputs can also start tests.



7. Mapping circuitry. Two 8-bit digital-to-analog converters provide oscilloscope mapping for a 16-bit address bus. This service aid can be built in or put on a separate board. The service manual must of course include maps of correctly operating systems.

isolate the board's data lines from the bus; it also removes the load supplied by the processor, and many switching power supplies will operate only with a full load. To compensate for this, the board needs another set of switches and some pullup resistors. With the switches, it is possible to select software designed to exercise only one board at a time.

Another way to isolate nodes is to operate the kernel in the free-run mode outside the instrument. To do this without an extender board, the serviceman may use jumpers to connect the supply, ground, and clock signal to the kernel. The designer can facilitate this procedure by providing easy access for the jumpers, such as test terminals.

The designer must provide the facility to open the feedback paths. If feedback is not broken when the signature-analysis program is sending a stimulus pattern to a device, then an error will propagate around the loop and all subsequent signatures will be wrong. It is a good idea to select an obvious feedback path to open, like the data bus of the kernel, where the added switches will interrupt the path.

Initialization of the boards under test is extremely important in signature analysis. Without initialization, consistent signatures cannot be guaranteed from one example of the product to another, or even in the same instrument from power-up to power-up. Some circuits initialize themselves with their own reset pulses after passing through one complete cycle. Otherwise, the designer must insure that the service manual documents an initialization procedure for the serviceman to follow.

Similarly, there must be provision for disabling asynchronous devices like monostables and interrupts. Also, multilayer boards should be avoided where there are bus lines going to several devices. It is difficult to use a current tracer with a multilayer board to discover which device is holding a bus line low or high.

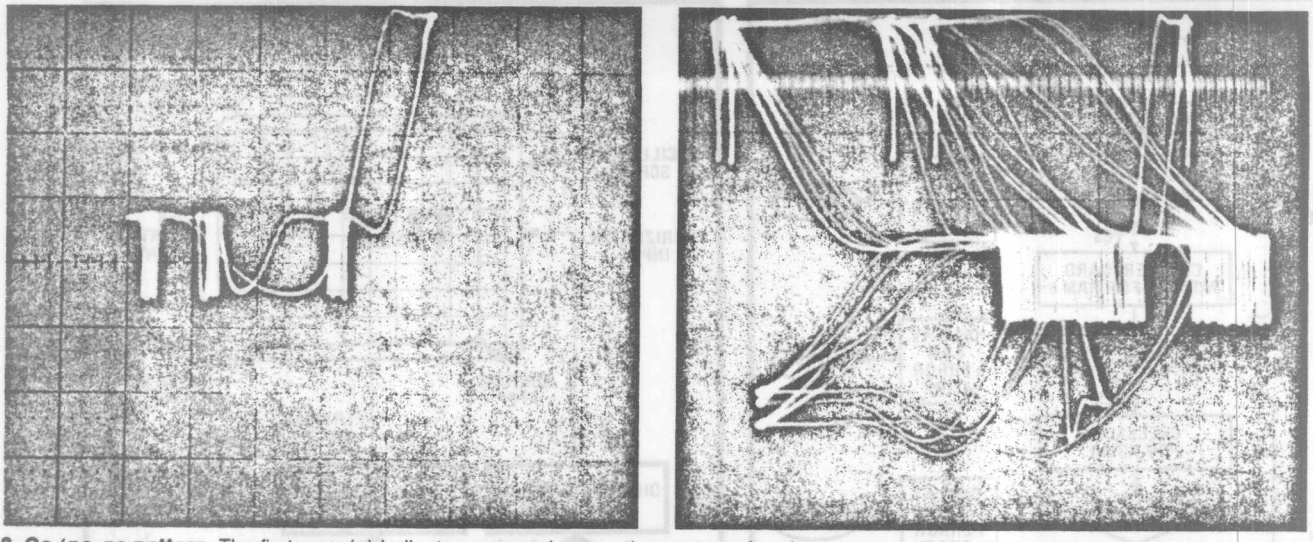
Spotting defective ROMs

The designer can facilitate isolation of a ROM failure by insuring that the address decoding circuitry is used to generate the measurement window for each chip. The chips shown added to the extender board in Fig. 5 perform this task. Then the start/stop signal for each ROM aligns with its first/last address, and the signatures appearing on the data bus come from only one device at a time (assuming that the output enables are working).

It also is important to provide for disconnection of the data bus when taking signatures on ROMs. In free-run analysis, other devices are being addressed and are putting out random data that results in erroneous signatures appearing on the bus lines.

Three-state devices can present a particular problem in obtaining stable, repeatable signatures. Nonrepeating signatures can occur if the start/stop window brackets the device's transition from high-impedance output to enable output when pullup resistors are absent.

To avoid this problem, pullup resistors should be on the board with the three-state device, not on some other board. Otherwise the extender board will need pullups to function when the board is isolated from the rest of the product. An alternative is to use address decoding that



8. Go/no-go pattern. The first map (a) indicates a properly operating system after the power-up diagnostic test. The second map (b) shows the same test on a similar system with the kernel malfunctioning. The serviceman checks the pattern, not individual points.

ensures the start/stop window only brackets the cycle time in which the three-state outputs are enabled.

Finally, it is up to the designer to provide the careful documentation of the signatures that will go into the service manual. Fortunately, there are several ways to insure that the signatures being read will be consistent from product to product.

The first step is to check that the signatures are stable and repeatable on several product samples.

The most stringent test features reducing the clock frequency by 50%. This procedure should leave the signatures unchanged—otherwise there is a potential timing problem, probably due to excessive bus settling time and usually caused by the lack of pullup resistors on the bus lines. To reduce the clock frequency, all that is necessary is to pull the kernel out of the unit and to use an external pulse generator as a clock instead of jumpering the clock from the unit under test.

Moreover, it is important always to document the characteristic 1s signature for each new setup so that the serviceman can determine if the start/stop signals supplied to the signature analyzer are correct. This characteristic signature is obtained by placing the data probe on a transistor-transistor-logic high level that remains high during the start/stop window.

Designing a microprocessor-based product to work with signature analyzers is a giant step towards serviceability. However, the designer can do more to help out field service, and one important consideration to take into account is the inclusion of diagnostic subroutines.

Built-in diagnostics

Product-initiated self tests and user-callable tests are the two basic kinds of built-in diagnostics. Self-initiated routines are automatically exercised by the product every time some preset condition is met. A user-callable routine is exercised only when the serviceman selects it.

Figure 6 is an example of the flow chart of a self-initiated test. A convenient implementation point for such diagnostics is power-on. When the processor fetches its power-on address in ROM, a pointer causes program execution to begin with a check of RAM memory. In this routine, a checkerboard 1 and 0 pattern is written into each RAM and then read back. If this pattern is not identical to the one sent, the processor initiates display of an error code. In the HP 5370A universal time-interval counter, for example, a display readout of *Err 6.1* means that RAM 1 failed the check, while *Err 6.2* means that RAM 2 failed, and so on.

If all the RAMs pass the test, a ROM checksum test is next. The contents at each ROM location are added up until the final checksum is compared with the correct one—in this case, the sum in the first location of each ROM. An incorrect checksum will cause an error message on the display. In the 5370, *Err 7.1* means ROM 1 failed the checksum test, and so on. (Of course, this routine works only if that portion of ROM containing the diagnostic program is good.)

If all ROMs pass their checksum tests, the self-diagnostic program advances to a display check. All segments of the light-emitting-diode display and all front-panel lamps are lit briefly. These are, of course, just a few examples of the many kinds of checks that can be performed at power-on.

User-callable diagnostics

There are two basic ways for a designer to implement user-callable diagnostics. Some routines can be initiated by pressing the appropriate front-panel keys. Others can be selected by setting internal servicing switches to the appropriate codes. There are many choices as to which diagnostic routines to include, with the choice depending on the type of product.

One generally useful diagnostic routine is the algorithm-tracing program. The product goes through its usual operating algorithm but also displays mnemonic codes at key points.

For example, in the HP 5342A microwave frequency counter, striking SET, SET, 0 on the keyboard initiates a diagnostic program display in four points in the algo-

Design for serviceability check list

Taking into account the following field-service considerations at the initial design stage will realize several advantages. Service calls are shortened, redesigns are eliminated, extra equipment is not needed, and the manufacturer and customer see their costs reduced.

1. Are inputs and outputs protected from normal abuse?
2. Are light-emitting diodes used to advantage inside the instrument to indicate proper operation? For instance, a lighted LED could indicate a locked phase-locked loop, clock is present, power-supply voltage OK, etc.
3. Are components located far enough from integrated circuits to allow test clips to be placed on the ICs?
4. Can feedback be easily disabled for troubleshooting purposes? A good example would be a jumper wire that can be removed to break feedback between several ICs on a board. Sometimes feedback can be broken by pulling a board in the feedback loop.
5. Are display LEDs easy to replace?
6. Are there power terminals for logic-probe/pulser/current-tracer operation?
7. Are interconnecting cables long enough to allow operation of boards when placed on extenders?
8. Are boards independent? Avoid, if possible, a device on one board and its pullup resistor on another. If this is necessary for proper line termination, provide on the board a switchable pullup resistor for stand-alone testing.
9. Is it possible to manually force a node to a particular state for troubleshooting purposes?
10. Can the instrument be operated with any of its circuit boards removed, i.e. does the power supply require a certain load?
11. Are boards keyed so that they cannot be inserted incorrectly?
12. Have all three types of troubleshooting documentation techniques been considered? These are:

- Symptom cause: list symptoms and possible causes. This technique can be helpful when used with microprocessors that can display error codes when a specific failure occurs.
- Troubleshooting tree: test and branch based on results of test. A good tree requires more development time than any other approach since its developer must convince himself that the person following a fault through the tree will not get lost in a wrong branch.
- Growing the kernel: the instrument is exercised in a series of operating modes arranged in increasing levels of complexity so that each successive operating mode exercises a larger percentage of the instrument. By observing the first operating mode in the sequence that fails, it is possible to quickly limit the problem to those assemblies that are used in the failed operating mode but are not used in the previous modes that all passed.

rithm's operation. At the beginning of the sweep portion, *SP* is displayed, indicating that the instrument is sweeping its internal synthesizer for the desired intermediate frequency signal. When the signal is detected, sweeping stops and a *2* is displayed. Then the intermediate frequency is centered in the passband, the display shows *3*. To indicate determination of the harmonic number (the harmonic of the synthesizer that is mixing with the unknown to produce the intermediate frequency), the display shows *Hd*.

Yet another useful user-callable diagnostic that applies to many products is a keyboard/display check. Such a test gives the serviceman confidence in using the keyboard and display in other diagnostic checks.

To initiate this test, the serviceman hits SET, SET, 8 on the keyboard of the 5342A. The next key pressed fills the display with a unique character associated with that key. For example, pressing the 1 key results in a display reading of all 1s, and so on.

A go/no-go test

A third testing technique that can simplify servicing of microprocessor-based products is mapping, which helps the serviceman decide where to begin troubleshooting. In the 5370A time-interval counter, for example, the first check is of the kernel and associated buses, for they must be working in order to troubleshoot any of the counting circuits, the display, and the analog sections.

Mapping can quickly check the operation of the microprocessor and its communication with other boards in the unit without spending time to take signatures. However, it is a go/no-go evaluation, so when it indicates a bad kernel, signature analysis or some other technique must be used to isolate the failed component.

Essentially, mapping is a picture of the address bus, and the processor generates the picture as it performs its routines. In the 5370A, rather than connect a logic-state analyzer to all 16 lines of the address bus, two digital-to-analog converters on a special service board plug into the bus to supply analog levels to the X and Y inputs of an oscilloscope (Fig. 7). Each dot of the address matrix displayed by the scope represents an individual address. As the processor performs its programmed routines, these dots are connected together to form a map.

The 5370A is designed so that power-on with no input signal results in the map of Fig. 8a. The map appears after the microprocessor executes its power-up diagnostics of RAM, ROM, etc. It indicates that the kernel is working properly, that the data and address buses are clear and functional, and that the counter is waiting for an input signal to start the instrument.

It is important to realize that the serviceman is not looking for each dot of the map; he is interested just in the pattern. There are only 12 maps for the 5370A that he needs to be able to recognize. Documented in the service manual, they include such routines as: measurement in progress, start signal but not stop, free-run, read/write test patterns, trigger-level routine, reset, and display-rate hold.

If the scope shows any other pattern (Fig. 8b), then the processor has entered an undefined routine or is stuck in a portion of the algorithm. If the cause is suspected to be a board holding a line on the address bus low or high, boards can be removed one at a time until the proper map appears on the scope. Otherwise, it's on to signature analysis. □

Design for serviceability check list

Taking into account the following field-service considerations at the initial design stage will realize several advantages. Service calls are shortened, redesigns are eliminated, extra equipment is not needed, and the manufacturer and customer see their costs reduced.

1. Are inputs and outputs protected from normal abuse?
2. Are light-emitting diodes used to advantage inside the instrument to indicate proper operation? For instance, a lighted LED could indicate a locked phase-locked loop, clock is present, power-supply voltage OK, etc.
3. Are components located far enough from integrated circuits to allow test clips to be placed on it's IC's?
4. Can feedback be easily disabled for troubleshooting purposes? A good example would be a jumper wire that can be removed to break feedback between several IC's on a board. Sometimes feedback can be broken by pulling a board in the feedback loop.
5. Are display LEDs easy to replace?
6. Are there power terminals for logic-probe/pulser/current-tracer operation?
7. Are interconnecting cables long enough to allow operation of boards when placed on extender?
8. Are boards independent? Avoid, if possible, a device on one board and its pullup resistor on another. If this is necessary for proper line termination, provide on the board a switchable pullup resistor for stand-alone testing.
9. Is it possible to manually force a node to a particular state for troubleshooting purposes?
10. Can the instrument be operated with any of its circuit boards removed, i.e. does the power supply require a certain load?
11. Are boards keyed so that they cannot be inserted incorrectly?
12. Have all three types of troubleshooting documentation techniques been considered? These are:
   - Symptom cause: list symptoms and possible causes. This technique can be helpful when used with microprocessors that can display error codes when a specific failure occurs.
   - Troubleshooting tree: test and branch based on results of test. A good tree requires more development time than any other approach since its developer must convince himself that the person following a fault through the tree will not get lost in a wrong branch.
   - Growing the kernel: the instrument is exercised in a series of operating modes arranged in increasing levels of complexity so that each successive operating mode exercises a larger percentage of the instrument. By observing the first operating mode in the sequence that fails, it is possible to quickly limit the problem to those assemblies that are used in the failed operating mode but are not used in the previous modes that all passed.

rithm's operation. At the beginning of the sweep portion, SP is displayed, indicating that the instrument is sweeping its internal synthesizer for the desired intermediate frequency signal. When the signal is detected, sweeping stops and a 2 is displayed. Then the intermediate frequency is centered in the passband, the display shows 3. To indicate determination of the harmonic number (the harmonic of the synthesizer that is mixing with the unknown to produce the intermediate frequency), the display shows 1f4.

Yet another useful user-callable diagnostic that applies to many products is a keyboard/display check. Such a test gives the serviceman confidence in using the keyboard and display in other diagnostic checks.

To initiate this test, the serviceman hits SET, SET, a on the keyboard of the 5442A. The next key pressed fills the display with a unique character associated with that key. For example, pressing the 1 key results in a display reading of all 1s, and so on.

A go/no-go test

A third testing technique that can simply servicing of microprocessor-based products is mapping, which helps the serviceman decide where to begin troubleshooting. In the 5370A time-interval counter, for example, the first check is of the kernel and associated buses, for they must be working in order to troubleshoot any of the counting circuits, the display, and the analog sections.

Mapping can quickly check the operation of the microprocessor and its communication with other boards in the unit without spending time to take signatures. However, it is a go/no-go evaluation, so when it indicates a bad kernel, signature analysis or some other technique must be used to isolate the failed component.

Essentially, mapping is a picture of the address bus, and the processor generates the picture as it performs its routines. In the 5370A, rather than connect a logic-state analyzer to all 16 lines of the address bus, two digital-to-analog converters on a special service board plug into the bus to supply analog levels to the X and Y inputs of an oscilloscope (Fig. 7). Each dot of the address matrix displayed by the scope represents an individual address. As the processor performs its programmed routines, these dots are connected together to form a map.

The 5370A is designed so that power-on with no input signal results in the map of Fig. 8a. The map appears after the microprocessor executes its power-up diagnostics of RAM, ROM, etc. It indicates that the kernel is working properly, that the data and address buses are clear and functional, and that the counter is waiting for an input signal to start the instrument.

It is important to realize that the serviceman is not looking for each dot of the map; he is interested just in the pattern. There are only 12 maps for the 5370A that he needs to be able to recognize. Documented in the service manual, they include such routines as: measurement in progress, start signal but not stop, free-run, read/write test patterns, trigger-level routine, reset, and display-rate hold.

If the scope shows any other pattern (Fig. 8b), then the processor has entered an undefined routine or is stuck in a portion of the algorithm. If the cause is suspected to be a board holding a line on the address bus low or high, boards can be removed one at a time until the proper map appears on the scope. Otherwise, it's on to signature analysis. □

Free-running signature analysis simplifies troubleshooting

Although the advent of μPs has made designing easier, it has also created a troubleshooting nightmare. Fortunately, a solution exists.

Andrew Stefanski, Hewlett-Packard Co

Failures that cause the complete breakdown of a μP-based product also generate extremely difficult troubleshooting tasks. But of all such challenges the location of a faulty component in the μP's bus structure is particularly troublesome. A failure there causes all components to behave abnormally because the bus forms a complex feedback loop with the microprocessor and memory. A fault anywhere on the bus propagates throughout this loop, resulting in erroneous inputs to each component and, as a consequence, erroneous outputs from each one. Logical in-circuit testing of such a μP-bus system gone wild is practically impossible.

But you can overcome feedback-loop-caused difficulties by eliminating the loop itself. A simple, easily constructed test fixture breaks the loop, enabling the processor to operate in a free-running mode—thus forcing it to behave predictably and allowing you to troubleshoot in an orderly fashion.

In addition to breaking the μP free from its data bus, this test fixture provides a no-operation (NOP) opcode to the bus so that the processor sees a NOP instruction regardless of the contents of the address being fetched. This instruction increments the program counter and causes a fetch of the next instruction (another NOP). Utilizing this technique forces the processor to address the entire memory-address space despite failures in the bus, address decoder or ROM. You can then verify μP operation step by step, checking the address lines with a signature-analyzer probe and comparing the

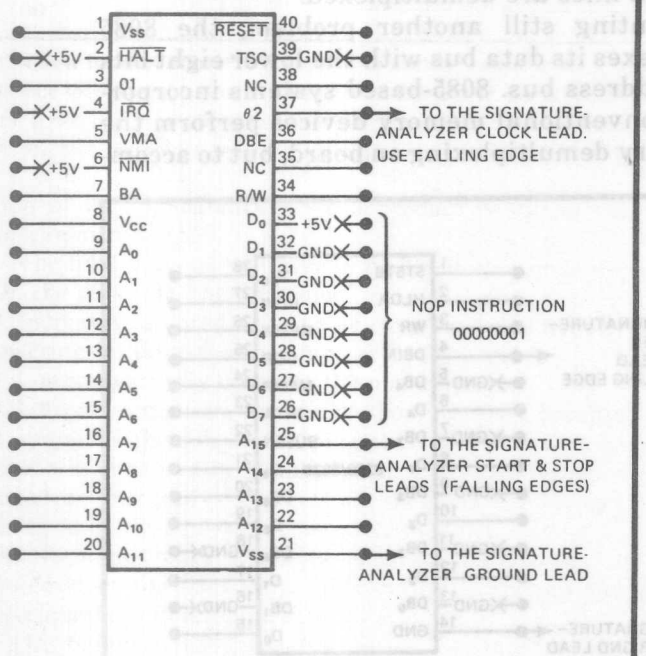


Fig 1—This 6800 free-run test fixture *provides a NOP instruction (0000001) to the data bus to step the microprocessor through its entire address space.*

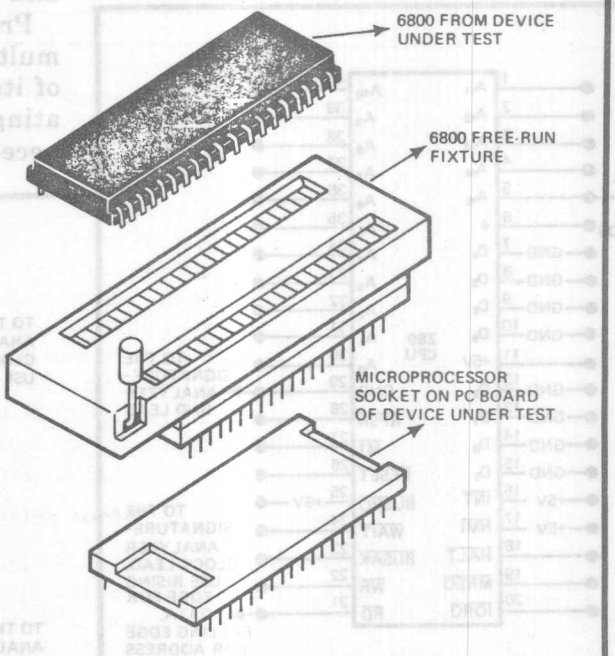


Fig 2—The test fixture *is inserted between the chip under test and its socket on the pc board.*

Reprinted from *EDN*, February 5, 1979.

A simply constructed test fixture lets a μP free-run

signatures displayed with correct signatures. Thus, you can utilize the signature analyzer to probe any portion of the circuit where correct inputs and outputs are known.

Preparation for this troubleshooting procedure involves inserting the free-run test fixture into a known good circuit and noting the signatures of all bus lines and nodes of the circuit's address decoders. The address-bus signatures for all μPs discussed in this article are the same (the notation—a modified hexadecimal scheme—is chosen for visual clarity):

A_0	UUUU	A_8	HC89
A_1	5555	A_9	2H70
A_2	CCCC	A_{10}	HPP0
A_3	7F7F	A_{11}	1293
A_4	5H21	A_{12}	HAP7
A_5	0AFA	A_{13}	3C96
A_6	UPFH	A_{14}	3827
A_7	52F8	A_{15}	755U.

In fact, this list is valid for *all* μPs with 16 address bits, so you can use it, in conjunction with the appropriate test fixture, to check μC boards fresh from the prototype shop—even if those boards have never been previously characterized. Misconnected traces, shorts and opens all appear quickly and easily.

Designing the fixture

Obviously, the key to implementing this approach is the design of the test fixture itself: It must provide a method of disconnecting the μP from its data bus and applying the NOP instruction. The exact fixture design required depends on the type of microprocessor used. The theory, however, applies universally.

The test fixture for the 6800 is particularly straightforward. **Fig 1** shows the wiring for a 40-pin socket that is inserted between the μP chip and its own socket (**Fig 2**). The data bus (pins 26 through 33) is tied to ground with the exception of D_0 (pin 33), which is connected to +5V; this approach provides a 00000001 NOP instruction. The ϕ_2 clock (pin 37) connects to the signature-analyzer clock lead—use the signal's falling edge for your trigger. V_{SS} (pin 21) provides signature-analyzer ground. Disconnecting the interrupt lines (pins 2, 4 and 6) and tying them to +5V prevents interference from interrupts during the test.

A fixture for the Z80 (**Fig 3**) utilizes similar techniques; the wiring simply changes to suit that processor's unique pinouts and input requirements for generating a NOP instruction. The 8080, however, requires a somewhat different approach because its status signals are multiplexed with the data; straightforward interruption of the data lines would thus completely prevent its operation. Therefore, the 8080 test-fixture approach (**Fig 4**) involves breaking the data lines on the "outside" side of an 8228/38 system controller chip—the side where the status and data lines are demultiplexed.

Presenting still another problem, the 8085 multiplexes its data bus with the lower eight bits of its address bus. 8085-based systems incorporating conventional memory devices perform the necessary demultiplexing on board, but to accom-

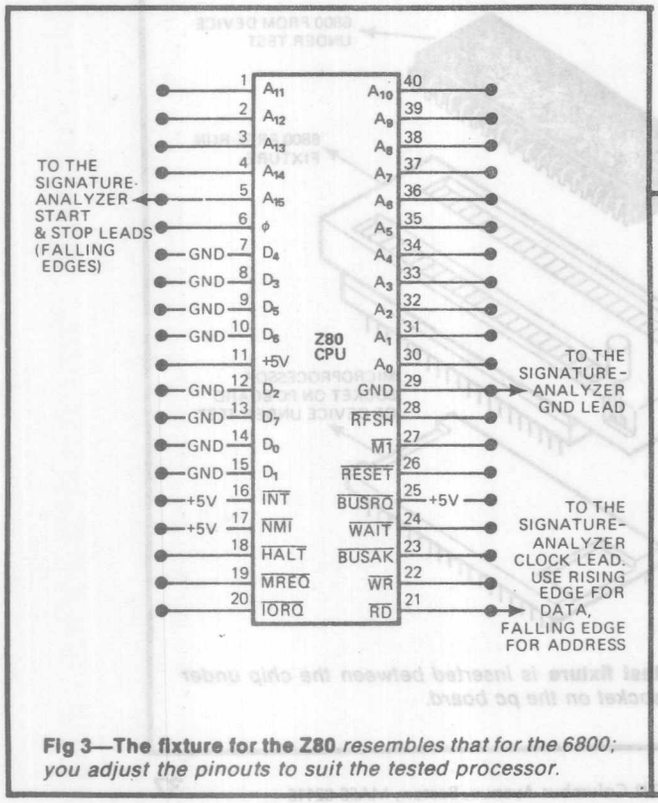


Fig 3—The fixture for the Z80 *resembles that for the 6800; you adjust the pinouts to suit the tested processor.*

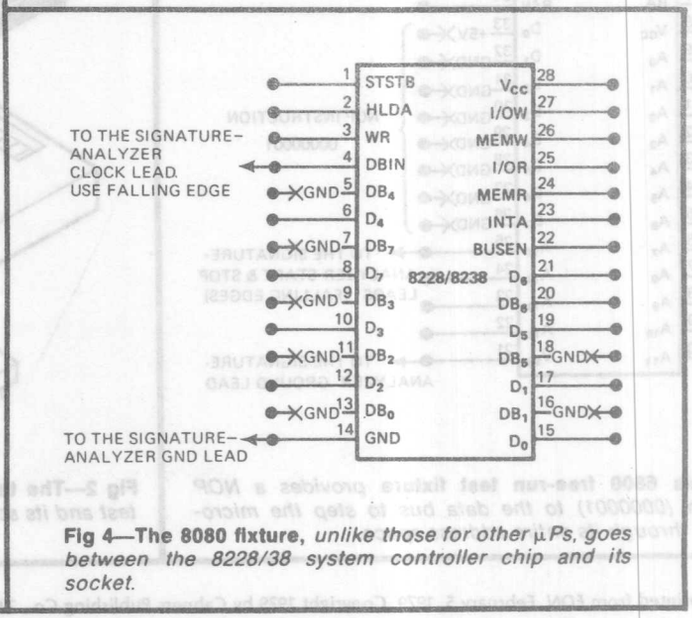


Fig 4—The 8080 fixture, *unlike those for other μPs, goes between the 8228/38 system controller chip and its socket.*

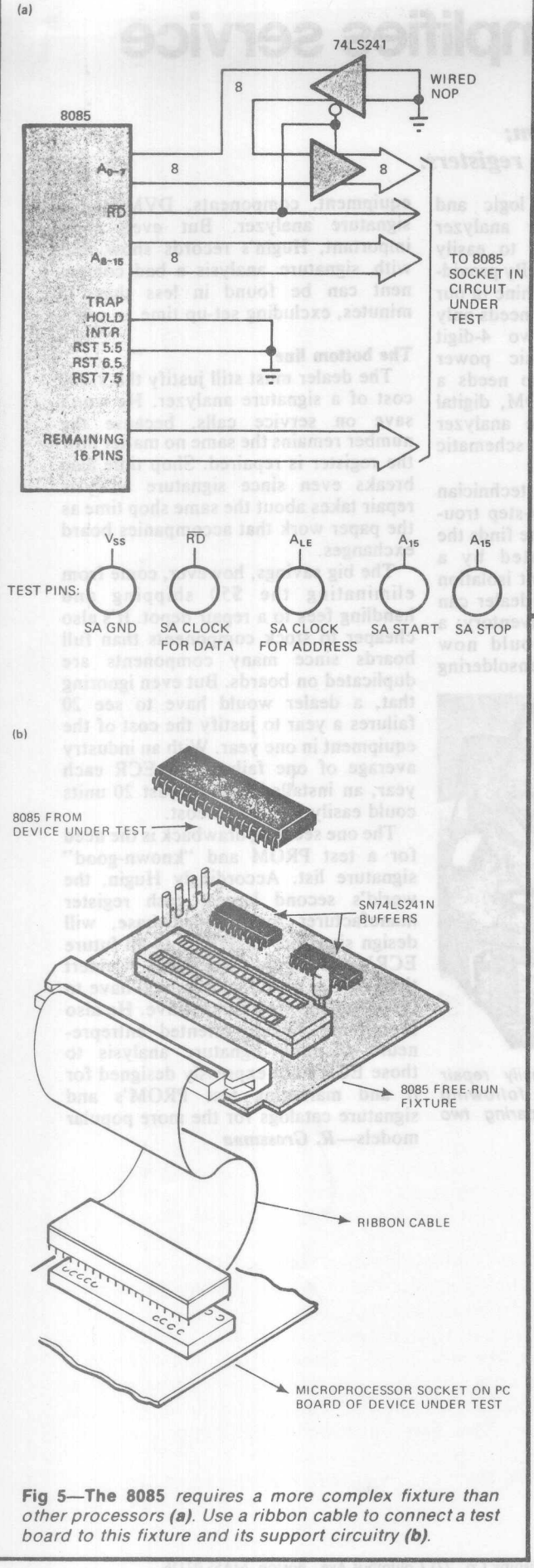


Fig 5—The 8085 *requires a more complex fixture than other processors* ***(a)****. Use a ribbon cable to connect a test board to this fixture and its support circuitry* ***(b)****.*

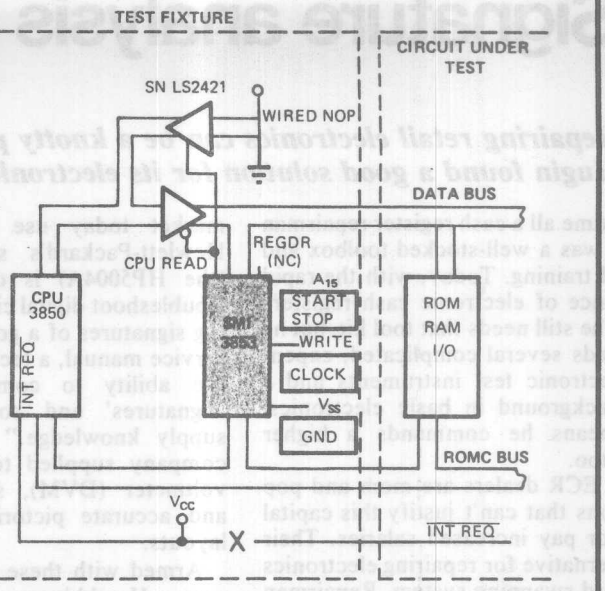


Fig 6—An F8 fixture *might require addition of a 3853 static-memory interface to generate Start and Stop signals for the signature analyzer.*

modate all possible systems, the 8085 free-run fixture uses a bidirectional buffer. This buffer preserves bus integrity when the μP sends an address out; it also breaks the bus and provides the NOP code when data is read in. Fig 5 shows a schematic of the free-run circuit for the 8085 and its implementation.

Finally, the F8 presents still another problem, because some F8-based systems don't have a visible address bus. Thus, if the tested board doesn't contain a static-memory interface chip (3853), you must incorporate one in the test fixture. Interfacing the 3850 CPU to the data bus also requires buffers so the CPU can use the bus to reset program counters when the system is powered up. The procedure for this μP remains relatively straightforward, however—one test board interfaced through a ribbon cable does the trick. **EDN**

Author's biography

Andrew Stefanski is a senior member of the IEEE and project leader at Hewlett-Packard's Santa Clara, CA Instrument Div. He holds a PhD EE from the University of Pennsylvania and an MSEE from Warsaw Polytechnic, Poland.

Signature analysis simplifies service

Repairing retail electronics can be a knotty problem; Hugin found a good solution for its electronic cash registers.

At one time all a cash register repairman needed was a well-stocked toolbox and minimal training. Today, with the rapid acceptance of electronic cash registers (ECR), he still needs that tool kit, but he also needs several complicated, expensive electronic test instruments and a solid background in basic electronics. That means he commands a higher salary, too.

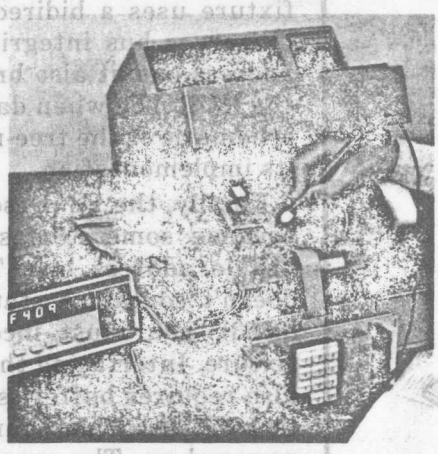
Most ECR dealers are mom and pop operations that can't justify this capital outlay or pay increased salaries. Their only alternative for repairing electronics is a board swapping system. Repairman plug known-good boards in until the ECR works and send the "bad" board to a factory depot for repairs.

With five days for postal handling each way and five days in the depot it takes a minimum of two weeks to get the board back. In addition to the actual repair charges, it costs an average of $50 for each board processed this way. Furthermore, the dealer has to stock spares for every board type, including new versions, as manufacturers improve and up their product lines.

The "electronic" tool kit

Cash register dealers are an independent breed; they want to make repairs themselves and they want to make them on-site. This type of market pressure convinced Hugin Kassaregister AB of Stockhold, Sweden to look for a low-cost, minimal-training electronic equivalent of the tool kit. It thinks it has found the answer in signature analysis. Says Jack E. Kleinert, US supervisor of service and education for Hugin Cash Registers, Inc., "All ECR's on the market today use digital logic and Hewlett-Packard's signature analyzer (the HP5004A) is designed to easily troubleshoot digital circuitry. By providing signatures of a good machine in our service manual, a technician needs only the ability to compare two 4-digit 'signatures' and some basic power supply knowledge." He also needs a company supplied test PROM, digital voltmeter (DVM), signature analyzer and accurate pictorial and schematic layouts.

Armed with these items a technician uses a Heathkit-type step-by-step troubleshooting procedure until he finds the faulty component, indicated by a "wrong" signature. With fault isolation at the component level the dealer can eliminate his spare board inventory; a complete repair center could now consist of soldering and unsoldering equipment, components, DVM and a signature analyzer. But even more important, Hugin's records show that with signature analysis a bad component can be found in less than 15 minutes, excluding set-up time.



***Semiskilled workers** can easily repair printed circuit boards by following pictorial diagrams and comparing two 4-digit signatures.*

The bottom line

The dealer must still justify the $1000 cost of a signature analyzer. He won't save on service calls, because the number remains the same no matter how the register is repaired. Shop time also breaks even since signature analysis repair takes about the same shop time as the paper work that accompanies board exchanges.

The big savings, however, come from eliminating the $50 shipping and handling fees to a repair depot. It's also cheaper to stock components than full boards since many components are duplicated on boards. But even ignoring that, a dealer would have to see 20 failures a year to justify the cost of the equipment in one year. With an industry average of one failure per ECR each year, an installed base of just 20 units could easily justify the cost.

The one seeming drawback is the need for a test PROM and "known-good" signature list. Accordingly Hugin, the world's second largest cash register manufacturer by installed base, will design signature analysis into all future ECR's and supply these items. Kleinert thinks other manufacturers will have to follow suit to stay competitive. He also sees some digitally-oriented entrepreneurs adapting signature analysis to those ECR's not expressly designed for it and marketing test PROM's and signature catalogs for the more popular models—***R. Grossman***